高职高专计算机任务驱动模式教材

PHP程序设计案例教程

何定华　主编　　周小松　刘超　黄治坤　副主编



清华大学出版社

高职高专计算机任务驱动模式教材

PHP程序设计案例教程

何定华　主编
周小松　刘超　黄治坤　副主编

清华大学出版社
北　京

内 容 简 介

本书从初学者角度出发，由浅入深、循序渐进地介绍了 PHP 和 MySQL 的一些知识，并提供了大量的 PHP 程序案例，课后还有习题供读者练习。

本书共分为 12 章，主要内容包括：PHP 概述与运行环境搭建、PHP 基本语法、PHP 数据处理、PHP 流程控制语句、PHP 函数、数组、字符串和正则表达式、面向对象的程序设计、PHP 表单应用、session 和 cookie、文件和目录处理、MySQL 数据库。每章后面都提供了一个综合案例及习题。

本书适合初学者使用。另外，对于大中专院校和培训班的学生，本书更是一本不可多得的教材。

图书在版编目(CIP)数据

PHP 程序设计案例教程/何定华主编. —北京：清华大学出版社，2019
(高职高专计算机任务驱动模式教材)
ISBN 978-7-302-51900-3

Ⅰ. ①P… Ⅱ. ①何… Ⅲ. ①PHP 语言—程序设计—高等职业教育—教材 Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字(2018)第 288149 号

责任编辑：张龙卿
封面设计：徐日强
责任校对：刘　静
责任印制：董　瑾

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　**邮　　编**：100084
社 总 机：010-62770175　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：三河市铭诚印务有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　**印　　张**：26　**字　　数**：627 千字
版　　次：2019 年 2 月第 1 版　**印　　次**：2019 年 2 月第1 次印刷
定　　价：59.80 元

产品编号：079519-01

编审委员会

出版说明

我国高职高专教育经过十几年的发展，已经转向深度教学改革阶段。教育部于2012年3月发布了教高〔2012〕第4号文件《关于全面提高高等教育质量的若干意见》，重点建设一批特色高职学校，大力推行工学结合，突出实践能力培养，全面提高高职高专教学质量。

清华大学出版社作为国内大学出版社的领跑者，为了进一步推动高职高专计算机专业教材的建设工作，适应高职高专院校计算机类人才培养的发展趋势，2012年秋季开始了切合新一轮教学改革的教材建设工作。该系列教材一经推出，就得到了很多高职院校的认可和选用，其中部分书籍的销售量超过了三四万册。现根据计算机技术发展及教改的需要，重新组织优秀作者对部分图书进行改版，并增加了一些新的图书品种。

目前，国内高职高专院校计算机相关专业的教材品种繁多，但符合国家计算机技术发展需要的技能型人才培养方案并能够自成体系的教材还不多。

我们组织国内对计算机相关专业人才培养模式有研究并且有过丰富的实践经验的高职高专院校进行了较长时间的研讨和调研，遴选出一批富有工程实践经验和教学经验的"双师型"教师，合力编写了该系列适用于高职高专计算机相关专业的教材。

本系列教材是以任务驱动、案例教学为核心，以项目开发为主线而编写的。我们研究分析了国内外先进职业教育的教改模式、教学方法和教材特色，消化吸收了很多优秀的经验和成果，以培养技术应用型人才为目标，以企业对人才的需要为依据，将基本技能培养和主流技术相结合，保证该系列教材重点突出、主次分明、结构合理、衔接紧凑。其中的每本教材都侧重于培养学生的实战操作能力，使学、思、练相结合，旨在通过项目实践，增强学生的职业能力，并将书本知识转化为专业技能。

一、教材编写思想

本系列教材以案例为中心，以技能培养为目标，围绕开发项目所用到的知识点进行讲解，并附上相关的例题来帮助读者加深理解。

在系列教材中采用了大量的案例，这些案例紧密地结合教材中介绍的各个知识点，内容循序渐进、由浅入深，在整体上体现了内容主导、实例解析、

以点带面的特点，配合课程采用以项目设计贯穿教学内容的教学模式。

二、丛书特色

本系列教材体现了工学结合的教改思想，充分结合目前的教改现状，突出项目式教学改革的成果，着重打造立体化精品教材。具体特色包括以下方面。

(1) 参照和吸纳国内外优秀计算机专业教材的编写思想，采用国内一线企业的实际项目或者任务，以保证该系列教材具有更强的实用性，并与理论内容有很强的关联性。

(2) 准确把握高职高专计算机相关专业人才的培养目标和特点。

(3) 每本教材都通过一个个的教学任务或者教学项目来实施教学，强调在做中学、学中做，重点突出技能的培养，并不断拓展学生解决问题的思路和方法，以便培养学生未来在就业岗位上的终身学习能力。

(4) 借鉴或采用项目驱动的教学方法和考核制度，突出计算机技术人才培养的先进性、实践性和应用性。

(5) 以案例为中心，以能力培养为目标，通过实际工作的例子来引入相关概念，尽量符合学生的认知规律。

(6) 为了便于教师授课和学生学习，清华大学出版社网站(www.tup.com.cn)免费提供教材的相关教学资源。

当前，高职高专教育正处于新一轮教学深度改革时期，从专业设置、课程体系建设到教材建设，依然有很多新课题值得我们不断研究。希望各高职高专院校在教学实践中积极提出本系列教材的意见和建议，并及时反馈给我们。清华大学出版社将对已出版的教材不断地进行修订并使之更加完善，以提高教材质量，完善教材服务体系，继续出版更多的高质量教材，从而为我国的职业教育贡献我们的微薄之力。

编审委员会

2017年3月

前　言

PHP(Hypertext Preprocessor,超文本预处理器)是一种通用开源脚本语言,其语法吸收了 C、Java 和 Perl 等语言的特点,应用广泛,主要适用于 Web 开发领域。PHP 独特的语法混合了 C、Java、Perl 以及 PHP 自创的语法。它可以比 CGI 或者 Perl 更快速地执行动态网页。用 PHP 做出的动态页面与其他的编程语言相比,PHP 是将程序嵌入 HTML(标准通用标记语言下的一个应用)文档中执行,执行效率比完全生成 HTML 标记的 CGI 要高许多;PHP 还可以执行编译后代码,编译可以加密和优化代码运行,使代码运行更快。

在众多 Web 开发技术中,PHP 拥有最多的使用者,它是开源的。PHP 简单易学,对初学者而言能够快速入门,专业程序员有许多的高级特性可以使用。

本书内容

本书分为 12 章,主要内容如下。

第 1 章讲述了 PHP 的基本概念、扩展库、Apache 和 IIS 服务器、PHP 运行环境的搭建。

第 2 章讲述了 PHP 的基本语法、PHP 注释和 HTML 注释以及 PHP 的输出。

第 3 章讲述了 PHP 标准数据类型(包括布尔型、整型和字符串型),以及复合数据类型(包括数组、对象等)。此外本章还讲述了数据类型的转换,变量、常量和运算符,最后还讲述了表达式。

第 4 章讲述了程序设计中的分支结构、循环结构,以及分支结构和循环结构的嵌套。

第 5 章讲述了 PHP 函数的概念、变量处理函数、数学函数、日期和时间函数及自定义函数、变量函数、嵌套函数和递归函数。

第 6 章讲述了数组的概念、数组的基本操作、数组的遍历、数组的排序和数组的其他操作。

第 7 章讲述了字符串的概念、字符串操作、字符串编码、正则表达式、正则表达式的验证和常用的 Web 验证。

第 8 章讲述了面向对象编程、类和对象的概念、类的成员、抽象类、final 的使用、实现类的特性、接口。

第 9 章讲述了表单的概念、表单提交和表单的高级操作。

第 10 章讲述了 session 的基本知识、session 的基本操作、session 举例、cookie 的基本知识、cookie 的基本操作。

第 11 章讲述了获取文件的属性、文件的基本操作、非线性读写文件、文件的高级操作、获取目录属性、目录的基本操作。

第 12 章讲述了 MySQL 数据库的概念、数据库以及数据表的创建、数据库服务器的连接、数据库的其他操作、结构化查询语句和数据查询等内容。

本书特色

（1）内容丰富、知识全面。本书全面细致地讲解了 PHP 相关知识，涵盖了 PHP 的基础知识及 MySQL 数据库的基础知识。

（2）零基础、通俗易懂。本书从初学者角度出发进行编写。不需要程序设计思想和其他前导课程，内容由浅入深，通俗易懂，是学习 PHP 程序设计的入门教程。

（3）案例丰富典型。为读者提供丰富、典型的案例，力求通过典型的案例把各个章节的知识点讲解透彻。本书讲授 PHP 函数较多，教师可以根据实际学时数进行适当删减。

（4）随书资源。为读者提供全书案例和教学 PPT。

读者对象

本书可以作为软件开发入门者的自学用书，也可以用作大中专相关专业的教学用书，还可以供开发人员查阅、参考。

本书由何定华、周小松、刘超、黄治坤编写。其中何定华担任主编，周小松、刘超、黄治坤担任副主编。全书由何定华审核并统稿。在本书的编写过程中，我们力求精益求精，但仍然存在一些不足之处，敬请读者提出宝贵意见，作者邮箱：hedinghua@qq.com。

编　者

2018 年 11 月

目　录

第 1 章　PHP 概述与运行环境搭建

知识点：

- PHP 擅长的领域
- PHP 的发展史
- PHP 的优点
- PHP 运行机制
- PHP 扩展库
- Apache 服务器和 IIS 服务器
- PHP 运行环境的搭建

本章导读：

PHP 是一种创建动态交互式站点的强有力的服务器端脚本语言，其代码可以直接嵌入 HTML 网页中，非常适合网站开发。在众多的 Web 开发技术中，PHP 以绝对优势拥有最多的使用者，它是一种开放源的脚本语言。PHP 语言简单、易学，对于初学者来说，能快速入门，而且为专业程序员提供了许多高级特性。

1.1　PHP 入门

PHP(Hypertext Preprocessor，超文本预处理器)是一种功能强大并且简便易用的脚本语言。其具有以下特点。

- PHP 是一种开源、跨平台、嵌入式的服务器端执行的动态网页开发语言。
- PHP 程序可以嵌入 HTML 内部。
- 与 Java 和C++ 不同，PHP 语言以基本语言为基础，语法简单。
- 支持多种主流和非主流的数据库，其中与 MySQL 是最佳组合。
- LAMP(Linux＋Apache＋MySQL＋PHP)是开发 Web 程序的最佳搭档。

PHP 擅长以下几个领域。

内容管理系统(CMS)：内容管理系统主要用于管理新闻、资料数据等，通常包含前台和后台程序。主要产品有 DedeCms，该软件是一个基于 MySQL 的数据库构建的文章管理系统，并且支持生产静态页面，适合于个人网站和一般商业网站的应用。其下载地址为 http://www.dedecms.com。

论坛系统(Forum)：论坛系统是一个支持用户间传递和共享信息的交流平台。论坛系统的功能相对复杂，要考虑到多人同时访问网站，其典型产品有“Discuz!”。“Discuz!”论坛是设计完善并可以适用于多种服务器环境的论坛系统。“Discuz!”在稳定性、负载、安全性

等方面都有不错的表现，其下载地址是 http://www.discuz.com。

电子商务系统(e-Bussiness)：电子商务系统是当前 Web 应用的一个很重要的应用。随着电子商务的发展，系统在安全性、功能设计方面都有很高的要求。典型产品有 ShopEx，该产品支持多种网上支付，界面美观，其下载地址是 http://www.shopex.cn。

1.1.1 PHP 的发展史

1. PHP/FI

PHP/FI 在 1995 年由 Rasmus Lerdorf 创建，最初只是一套简单的 Perl 脚本，用来跟踪访问它主页的人们的信息。它给这一套脚本取名为 Personal Home Page Tools。随着更多功能需求的增加，Rasmus 写了一个更大的 C 语言的实现程序，可以访问数据库，可以让用户开发简单的动态 Web 程序。Rasmus 发布了 PHP/FI 的源代码，以便每个人都可以使用它，同时大家也可以修正它的 Bug 并且改进它的源代码。

PHP/FI 是一个专为个人主页/表单提供解释程序的程序，已经包含了今天 PHP 的一些基本功能。它有着 Perl 样式的变量，自动解释表单变量，并可以嵌入 HTML，其语法本身与 Perl 很相似，但是它的功能很有限，很简单，还稍微有些不协调。

2. PHP/FI 2.0

到 1997 年，PHP/FI 2.0，也就是它的 C 语言实现的第二版在全世界已经有几千个用户和大约 50000 个域名安装，大约是 Internet 所有域名的 1%。但是那时只有几个人在为该工程撰写少量代码，它仍然只是一个人的工程。

PHP/FI 2.0 在经历了数个 beta 版本的发布后，于 1997 年 11 月发布了官方正式版本。不久，PHP 3.0 的第一个 alpha 版本发布，PHP 从此走向了成功。

3. PHP 3.0

PHP 3.0 是类似于当今 PHP 语法结构的第一个版本。Andi Gutmans 和 Zeev Suraski 在为一所大学的项目中开发电子商务程序时发现 PHP/FI 2.0 功能明显不足，于是他们重写了代码，这就是 PHP 3.0。经过 Andi、Rasmus 和 Zeev 一系列的努力，考虑到 PHP/FI 已存在的用户群，他们决定联合发布 PHP 3.0 作为 PHP/FI 2.0 的官方后继版本。而 PHP/FI 2.0 的进一步开发几乎终止了。

PHP 3.0 的一个最强大的功能是它的可扩展性。除了给最终用户提供数据库、协议和 API 的基础结构，它的可扩展性还吸引了大量的开发人员加入并提交新的模块。后来证实，这是 PHP 3.0 取得巨大成功的关键。PHP 3.0 中的其他关键功能包括面向对象的支持，以及更强大和协调的语法结构。

这个全新的语言伴随着一个新的名称发布。它从 PHP/FI 2.0 的名称中移去了暗含“本语言只限于个人使用”的部分，它被命名为简单的缩写 PHP。这是一种递归的缩写，它的全称是——Hypertext Preprocessor。

1998 年末，PHP 的安装人数接近 10000，有大约 100000 个网站报告自己使用了 PHP。在 PHP 3.0 的顶峰时期，Internet 上 10% 的 Web 服务器上都安装了它。

经过约 9 个月的公开测试后，官方于 1998 年 6 月正式发布 PHP 3.0。

4. PHP 4.0

1998 年的冬天，官方发布 PHP 3.0 不久，Andi Gutmans 和 Zeev Suraski 开始重新编

写 PHP 代码,设计目标是增强复杂程序运行时的性能和 PHP 自身代码的模块性。PHP 3.0 的新功能和广泛的第三方数据库、API 的支持,使这样的程序的编写成为可能,但是 PHP 3.0 没有高效处理如此复杂程序的能力。

新的被称为 Zend Engine(这是 Zeev 和 Andi 的缩写)的引擎,成功地实现了设计目标,并在 1999 年中期首次引入 PHP。基于该引擎并结合了更多新功能的 PHP 4.0,在 PHP 3.0 发布两年后,于 2000 年 5 月发布了官方正式版本。除了更高的性能以外,PHP 4.0 还包含了其他一些关键功能,比如,支持更多的 Web 服务器;支持 HTTP Sessions;有输出缓存(output buffering);更安全地处理用户输入的方法;一些新的语言结构。

5. PHP 5.0

虽然 PHP 4.0 已经能够胜任绝大多数的 Web 应用,但是事实上仍然有许多开发者抱怨,由于 PHP 4.0 在面向对象机制方面的欠缺,致使在开发大型企业级应用方面表现不足,这些专业领域仿佛早已经成为 C++/.NET/Java 的天下。

目前大部分的应用开发(尤其是互联网开发)需要将网站模式与理念实现为产品和应用,另外也可能需要在操作系统的后台实现复杂服务的功能。PHP 已经可以满足这一切的功能,能够使开发者快乐地工作。

PHP 5.0 是一个专业和高效率的开发工具,对于中小型项目,可以实现快速开发和性能优异的目标。而 PHP 5.0 在今后的市场定位,也并非扮演与 Java 或者 ASP.NET 竞争市场的角色。相信不远的将来,随着新的 PHP 7.0 的发布,PHP 的主流应用会有质的飞跃。

6. PHP 7.0

PHP 7.0 使用新版的 ZendEngine 引擎,带来了许多新的特性,以下是部分新特性。

- PHP 7.0 比 PHP 5.6 性能提升了两倍。
- 全面的 64 位支持。
- 以前的许多致命错误,现在改成抛出异常。
- 移除了一些老的不再支持的 SAPI(服务器端应用编程端口)和扩展。
- 新增加了结合比较运算符。
- 新增加了函数的返回类型声明。
- 新增加了标量类型声明。
- 新增加了匿名类。

1.1.2 PHP 的优点

PHP 安全性高、运行稳定,有活跃的用户群和广大的开发者社区,容易且适合用户快速学习,这些特点都可以是用户选择 PHP 开发 Web 应用程序的理由。PHP 的主要优点如下。

- 服务器端脚本。
- 命令行 Shell 脚本。
- 客户端用户界面。使用 PHP GTK 可以编写类似于 Visual C 开发的桌面级应用程序。
- PHP 是性能优越的编译程序,又具备解析过程的优点,是动态语言的典型代表。

• 开发效率高，函数语句简单明了。
• 输出控制灵活，可以在 HTML 中内嵌 PHP 脚本代码，也可以由 PHP 输出到 HTML 中运行；或者在命令行下执行，将结果输出到其他设备。
• 可以实现模板化，实现程序逻辑与用户界面的分离。
• 跨平台，开发的程序可以运行在不同的操作系统上。
• 与多个服务器（例如，Apache、Microsoft IIS 和 LightHttpd）兼容。
• 完全支持面向对象开发并向下兼容，支持过程与面向对象两种风格的开发。
• 内嵌 Zend 加速引擎，性能快速稳定。
• PHP 编写起来很容易，内置函数丰富，几乎包含了 Web 开发的所有方面。
• 组件化开发，提供 MySQL、PostgreSQL、Oracle 和 Microsoft SQL Server 等多种数据库系统的访问接口，支持 ODBC（Open Database Connectivity，开放数据库连接）。
• 扩展性好，支持访问 Win 32 系统的 COM 对象。
• 支持正则表达式，内置 POSIX 与 Perl，兼容两类的正则表达式支持。
• 开发成本低，工具多，且有众多使用 PHP 开发的源代码项目供开发者参考和二次开发。
• 支持利用通用的 MVC（Model-View-Controller）框架开发，在 PHP 中可以使用 Zend Framework、CakePHP、CodeLgniter 和 Symfony 等多个应用框架。
• 支持桌面级系统开发。PHP 不仅能够开发动态网站系统，还能够开发桌面级应用程序，以及 Shell 或命令行下运行的 Deamon 守护脚本以及服务器端管理程序。
• 支持加密分发代码。使用 Java 和.NET 这些虚拟机字节码的语言，在某些时候非常容易被反编译，从而导致一些安全问题，使用 Zend Optimizer 不仅可以使 PHP 实现“编译”运行和速度的飞跃，而且可以实现 PHP 源代码的完全加密，从而保护作者的利益以及软件版权。

1.1.3 PHP 的运行机制

图 1-1 是 PHP 的运行机制。首先是浏览器通过 Internet 向服务器发出 URL 请求；服务器收到来自浏览器的 URL 请求之后进行处理，并把结果嵌入 HTML 文档，再发送到浏览器（如果需要调用数据库，则会执行程序来查询数据库并返回数据）；浏览器收到来自服务器的 HTML 文档，将其显示出来。

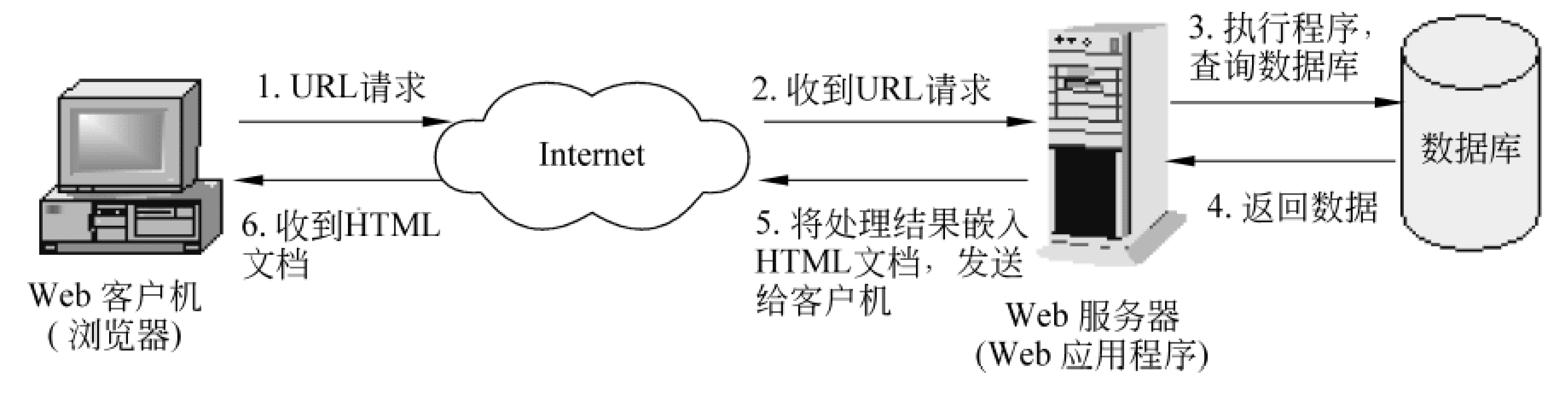


图 1-1 PHP 的运行机制

从图 1-1 中可以看出 PHP 是运行在服务器端的，浏览器负责 URL 请求和显示来自服务器的 HTML 文档（即 PHP 在服务器上执行的结果）。

1.2　PHP 扩展库

PHP 5.0 中的扩展库包括标准扩展库（Standard PHP Library，SPL）和外部扩展库（Extension Community Library，ECL）。

1.2.1　标准扩展库

标准扩展库就是被编译到 PHP 内部的库，历史上标准扩展库是指 Standard 扩展（默认编译进 PHP），但是 PHP 5.0 出现后，标准扩展库实际上成了代名词。PHP 5.0 新增了内置标准扩展库，类似 MySQL、MySQLi 和 GD2 等这些库则被放在外部扩展库中，需要时在 php.ini 配置文件中选择加载。

1. XML 扩展

在 PHP 5.0 中，所有的 XML（可扩展标记语言）扩展都已经被重写，libxml2 中的 XML 工具包为 PHP 的 XML 操作提供了更多的可维护性技术。

（1）DOM

虽然 PHP 4.0 版本已经使用 libxml2 库对 DOM（文件对象模型）提供支持，但是存在内存泄露的 Bug，并且 libxml2 对 W3C 也不太兼容。PHP 5.0 已经完全支持 DOM 扩展，不仅扩展库已重写，而且也被集成到 PHP 中，还对 W3C 标准全面兼容。

（2）SimpleXML

现在 PHP 开发者在操作使用 XML 时，又多了一种选择——SimpleXML，这是 PHP 5.0 全新开发的外部扩展库，用以代替 DOM 或较难用的 SAX（Simple API for XML）。

（3）XML Reader

PHP 5.0 中引入了新的 XMLReader 类，该类用于读取 XML 文件。它与 DOM、SimpleXML 不同的是：XMLReader 以流模式进行操作，即从头到尾读取文档，在文件后面的内容编译完成之前，可以先处理编译好的文件前面的内容，从而可以快速、高效地使用内存。

（4）SOAP 扩展

SOAP（Simple Object Access Protocol）是 Web 服务的一种简单对象访问协议，用于解决异构系统之间的信息传递问题。PHP 5.0 中已经完全重写了对 SOAP 支持的扩展库，而无须使用其他比较笨重的类库。

2. PHP 标准库

PHP 标准库是为解决一些在 Web 开发中普遍存在的问题，在 PHP 5.0 中提供一系列的接口和类的类库。它为开发者提供了迭代器、数组对象、运行时异常处理和观察者模式等数据容器与解决方案。

3. MySQLi 扩展

PHP 5.0 重新编写了一套新的 MySQL 数据库扩展库——MySQLi，它提供了新的特性以及针对新版本 MySQL 5.x 的优化功能。这个扩展提供了两个接口，分别用于面向过程和面向对象，PHP 开发者可以选择其中一种进行开发。

MySQLi扩展还支持预执行、变量绑定以及SSL(Secure Sockets Layer)连接支持、数据压缩连接、事务控制等实用的功能和方法。

4. PDO数据库抽象层

PDO(PHP Data Object)是PHP 5.1中新加入的数据库抽象层,直接预装在PHP的扩展中,它主要是为了解决访问不同数据库统一接口的问题。PDO是随着PHP 5.1软件包正式分发的,目前已经支持绝大多数的主流数据库系统(如MySQL和Oracle)。

5. JSON扩展

JSON(JavaScript Object Notation)是一种非常轻量级的数据交换格式,主要用于JavaScript与服务器端PHP脚本的交互。从PHP 5.2版本起,它正式在内部集成JSON功能。

1.2.2 外部扩展库

PHP 5.0中的外部扩展库主要包含两方面的扩展。

1. PECL扩展

PECL(PHP Extension Community Library,PHP社区扩展库)与官方PHP开发小组和发布的扩展库相似,都作为PHP的扩展库。PECL以C/C++作为底层语言开发并在PHP社区上发布,不需要PHP开发小组认证。

2. PEAR扩展

PEAR(PHP Extension and Application Repository,PHP扩展与应用库)是PHP代码库,是标准程序功能提供用纯粹的PHP代码预先编写的类,它包含有丰富特性的功能类库,如数据库、邮件、时间和错误处理等,使用PEAR会大大降低应用程序的开发时间。

1.3 Web服务器

PHP支持多个Web服务器,其中最常用的服务器是Apache和IIS服务器。除此之外,它还可以稳定运行在LightHttpd、Netscape、WebServer和Zenus等非主流Web服务器上。其中黄金搭档是Apache服务器。

1.3.1 Apache服务器

Apache HTTP Server(简称Apache)是Apache软件基金会的一个开放源码的网页服务器,可以在大多数计算机操作系统中运行,由于其多平台和安全性被广泛使用,是最流行的Web服务器端软件之一。它快速、可靠并且可通过简单的API扩展,将Perl/Python等解释器编译到服务器中。Apache HTTP服务器是一个模块化的服务器,源于NCSAhttpd服务器,经过多次修改,成为世界使用排名第一的Web服务器软件。它可以运行在几乎所有广泛使用的计算机平台上。

Apache源于NCSAhttpd服务器,经过多次修改,成为世界上最流行的Web服务器软件之一。Apache取自a patchy server的读音,意思是充满补丁的服务器,因为它是自由软件,所以不断有人来为它开发新的功能、新的特性,并修改原来的缺陷。Apache的特点是简

单、速度快、性能稳定,并可作为代理服务器来使用。

Apacheweb 服务器软件拥有以下特性。

- 支持最新的 HTTP/1.1 通信协议。
- 拥有简单而强有力的基于文件的配置过程。
- 支持通用网关接口。
- 支持基于 IP 和基于域名的虚拟主机。
- 支持多种方式的 HTTP 认证。
- 集成 Perl 处理模块。
- 集成代理服务器模块。
- 支持实时监视服务器状态和定制服务器日志。
- 支持服务器端包含指令(SSI)。
- 支持安全 Socket 层(SSL)。
- 提供用户会话过程的跟踪。
- 支持 FastCGI。
- 通过第三方模块可以支持 JavaServlets。

如果你准备选择 Web 服务器,毫无疑问 Apache 是你的最佳选择。

1.3.2　IIS 服务器

IIS 是 Internet Information Server 的缩写,它是微软公司主推的服务器。IIS 与 Window NT Server 完全集成在一起,因而用户能够利用 Windows NT Server 和 NTFS (NT File System,NT 的文件系统)内置的安全特性,建立强大、灵活而安全的 Internet 和 Intranet 站点。

IIS 支持 HTTP(Hypertext Transfer Protocol,超文本传输协议)、FTP(File Transfer Protocol,文件传输协议)以及 SMTP 协议,通过使用 CGI 和 ISAPI,IIS 可以得到高度的扩展。

IIS 支持与语言无关的脚本编写和组件,通过 IIS,开发人员就可以开发新一代动态的、富有魅力的 Web 站点。IIS 不需要开发人员学习新的脚本语言或者编译应用程序,IIS 完全支持 VBScript、JScript 开发软件以及 Java,它也支持 CGI 和 WinCGI,以及 ISAPI 扩展和过滤器。IIS 也能很好地支持 PHP。

1.4　PHP 运行环境的搭建

要想在自己的计算机上开发与运行 PHP,需要搭建 PHP 运行环境。目前一般会把运行 PHP 的各个软件集成在一起,安装起来非常方便,甚至连配置都不需要。这些软件有 XAMPP、AppServ、phpStudy 等。本书使用 phpStudy,目前最新版本是 phpStudy 2016。

phpStudy 包集成最新的 Apache+PHP+MySQL+phpMyAdmin,一次性安装,无须配置即可使用,是非常方便、好用的 PHP 运行和调试环境。该程序不仅包括 PHP 运行和调试环境,还包括 PHP 开发手册等。总之学习 PHP 只需一个包。

对学习 PHP 新手来说,Windows 下环境配置是一件很困难的事;对老手来说也是一件

烦琐的事。因此无论你是新手还是老手,该程序包都是一个不错的选择。

phpStudy 2016 下载地址为 http://www.phpstudy.net/phpstudy/phpStudy20161103.zip。下载之后解压到一个文件夹中。为了能够正常地运行 PHP 等软件,用户的计算机还需要安装 Visual C++ 2008 运行库。32 位的 Visual C++ 2008 运行库下载地址: http://www.microsoft.com/zh-CN/download/details.aspx?id=5582;64 位的 Visual C++ 2008 运行库下载地址: http://www.microsoft.com/zh-CN/download/details.aspx?id=15336。

下面进行 phpStudy 2016 的安装。

步骤 1 下载最新的 phpStudy,解压到一个文件夹中,如图 1-2 所示。

步骤 2 双击 phpStudy20161103.exe 图标进行安装,如图 1-3 所示。

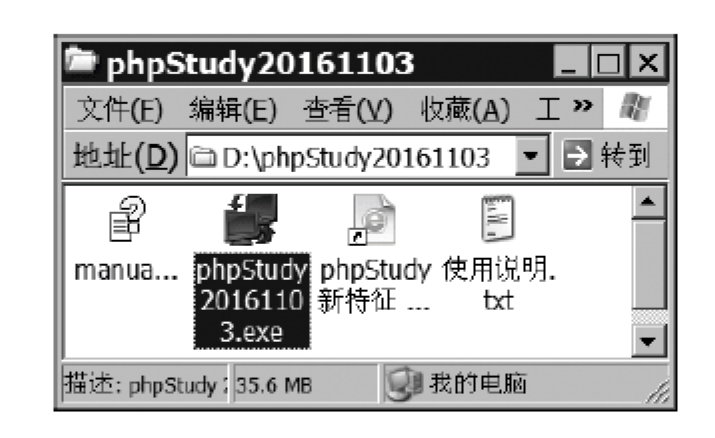


图 1-2 phpStudy 2016 软件



图 1-3 安装 phpStudy

步骤 3 选择安装的文件夹。此处不修改安装路径,直接单击 OK 按钮进入安装,过程如图 1-4 所示。

步骤 4 安装完毕,为了防止重复初始化,单击"是"按钮完成安装,如图 1-5 所示。

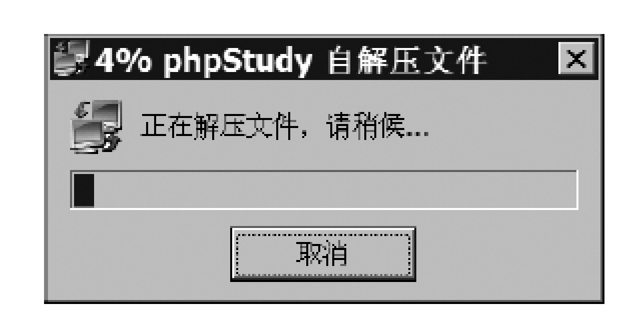


图 1-4 phpStudy 的安装过程

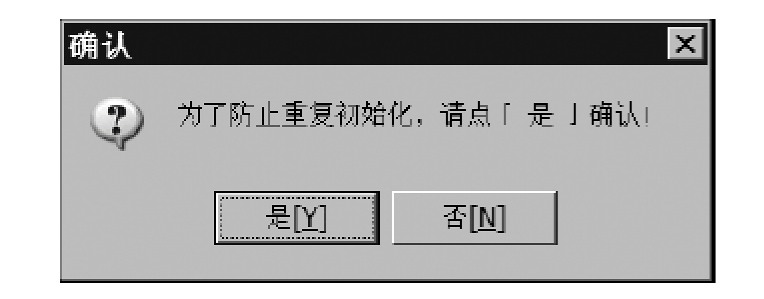


图 1-5 单击"是"按钮完成安装

步骤 5 下载 PHP 等软件运行所需要的 Visual C++ 2008 运行库。本书使用 Window XP,下载 32 位 Visual C++ 2008 运行库,文件名为 vcredist_x86.exe。为了支持 PHP 等软件的运行,需要安装 Visual C++ 2008 运行库。双击 vcredist_x86.exe 文件的图标运行 Visual C++ 2008 运行库,如图 1-6～图 1-10 所示。

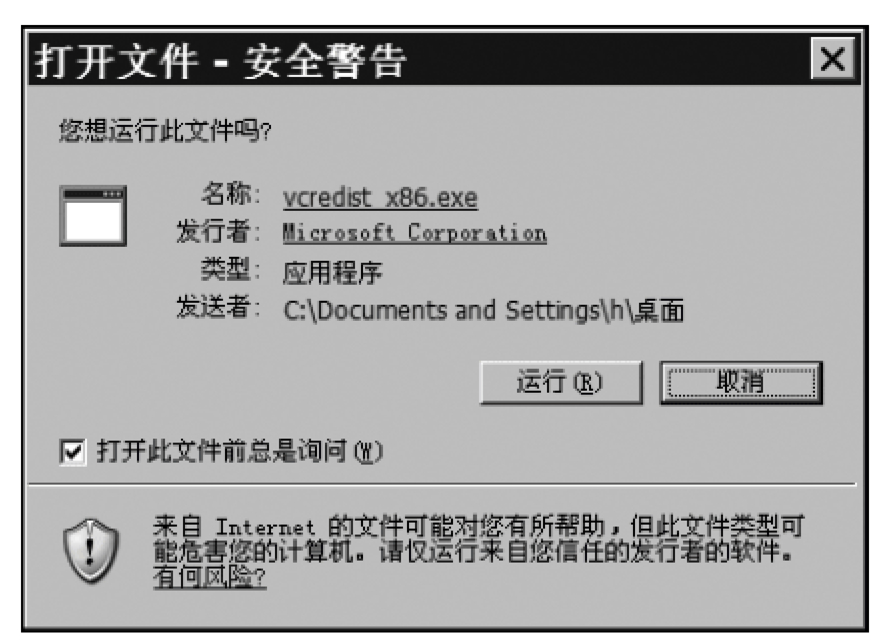


图 1-6 安装 Visual C++ 2008 运行库

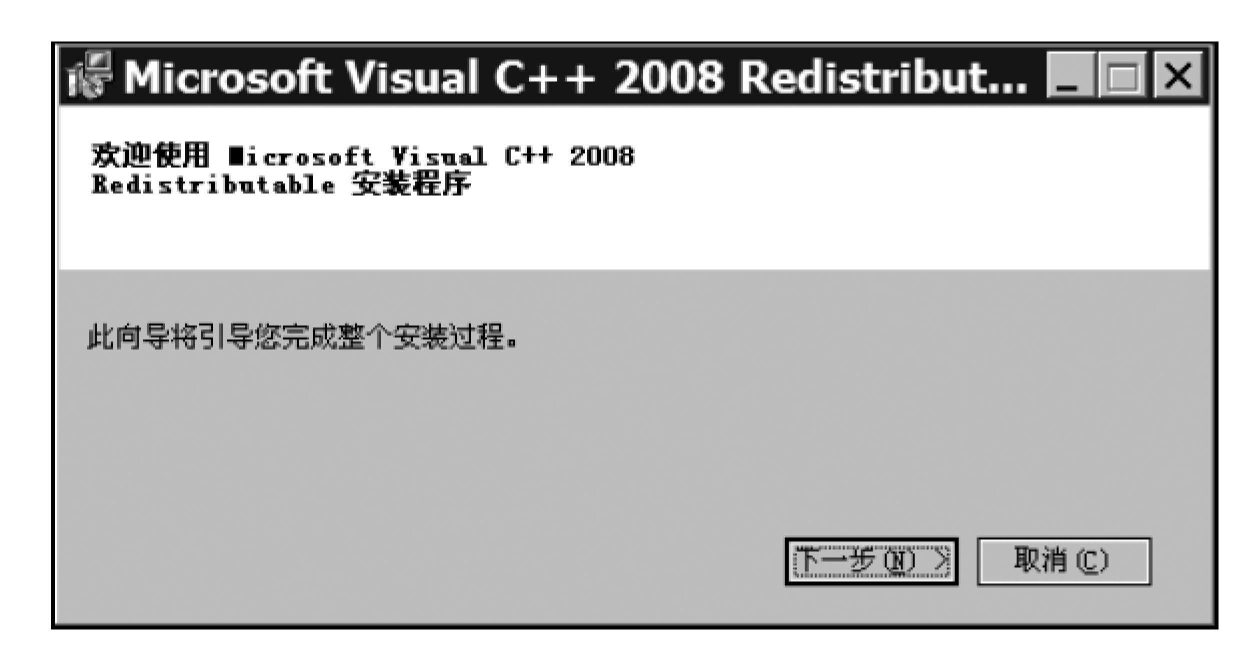


图 1-7　单击“下一步”按钮

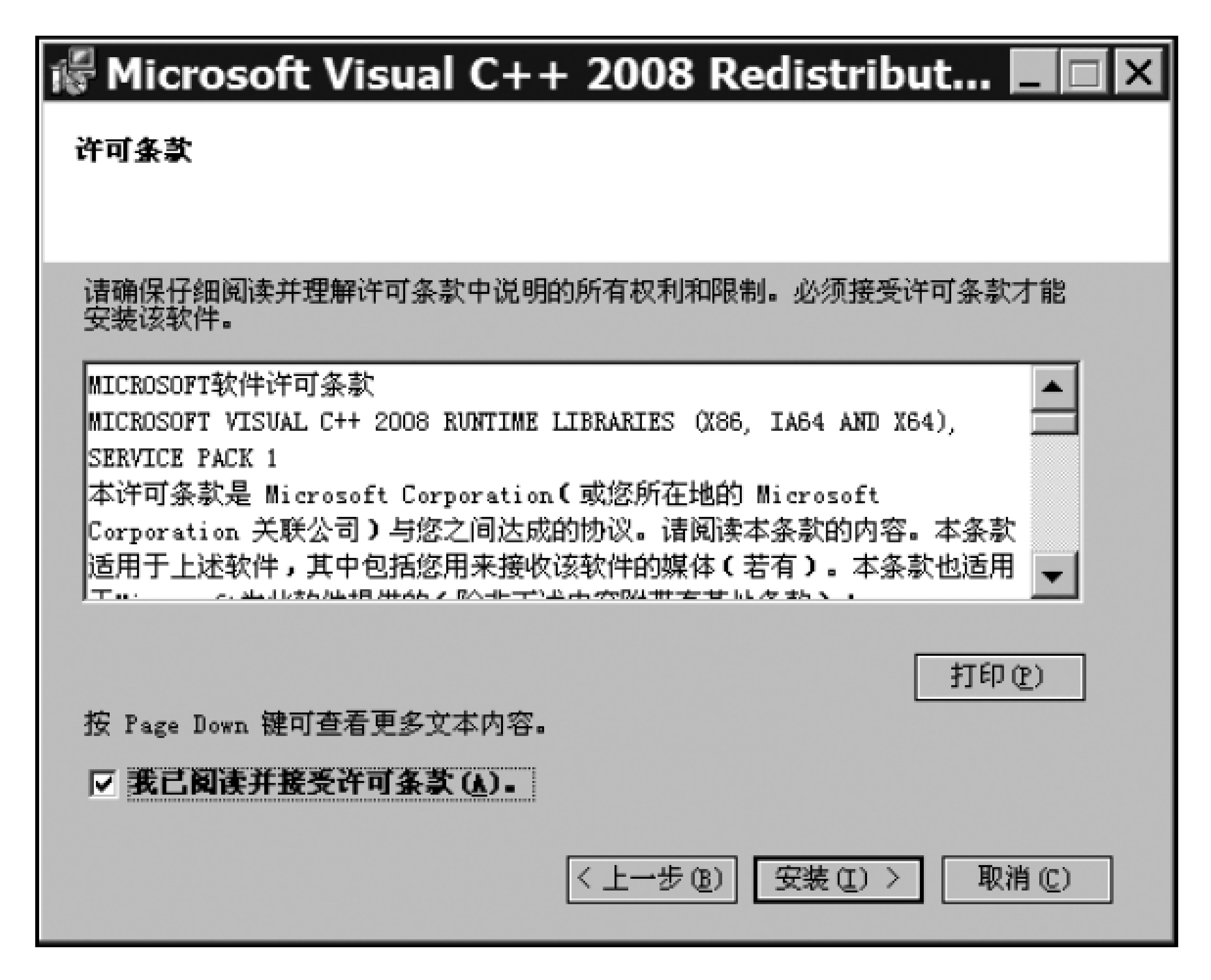


图 1-8　选中“我已阅读并接受许可条款”并单击“安装”按钮

图 1-9　安装过程

PHP 运行环境搭建完成之后的文件结构如图 1-11 所示。

其中，Apache 文件夹中存放了 Apache 服务器程序；IIS 文件夹中存放了 IIS 服务器程序；WWW 文件夹是 Apache 或者 IIS 网站的根目录，所有的网页文件都需要放置在该文件

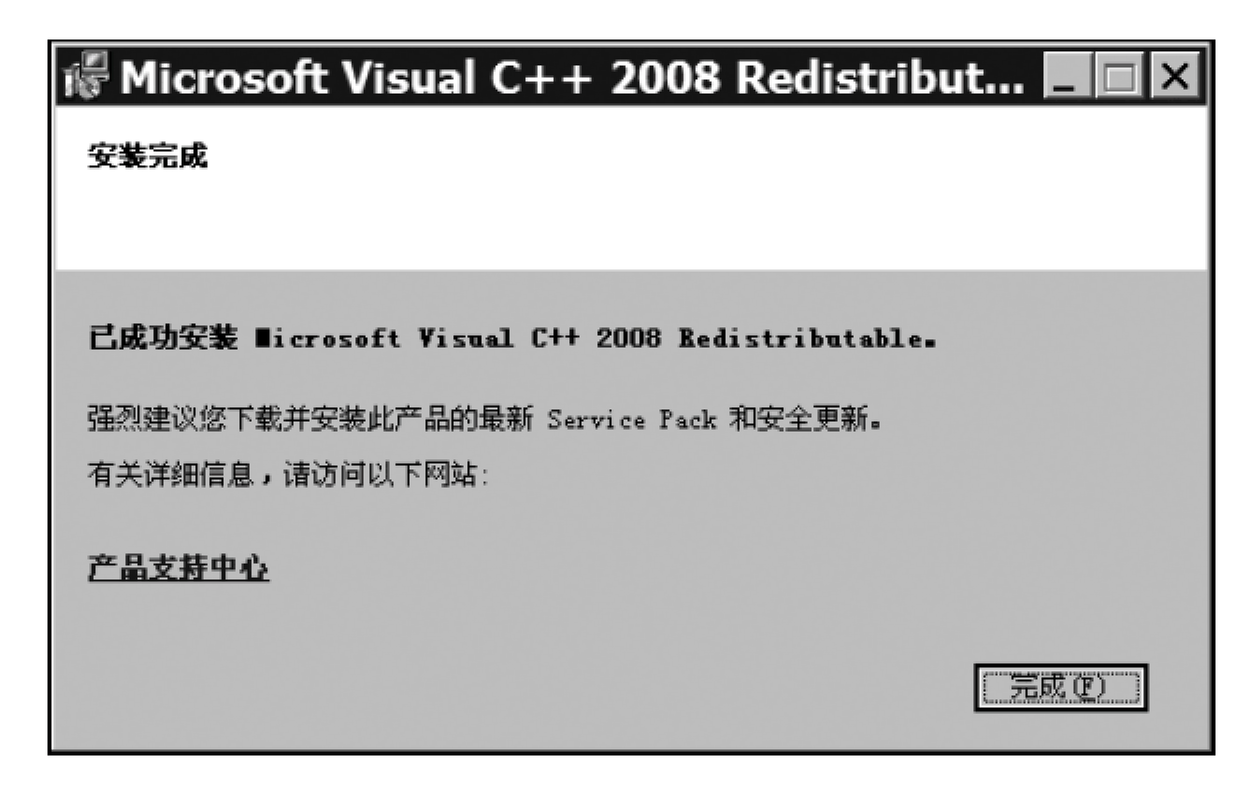


图 1-10　单击“完成”按钮完成 Visual C++ 2008 运行库的安装

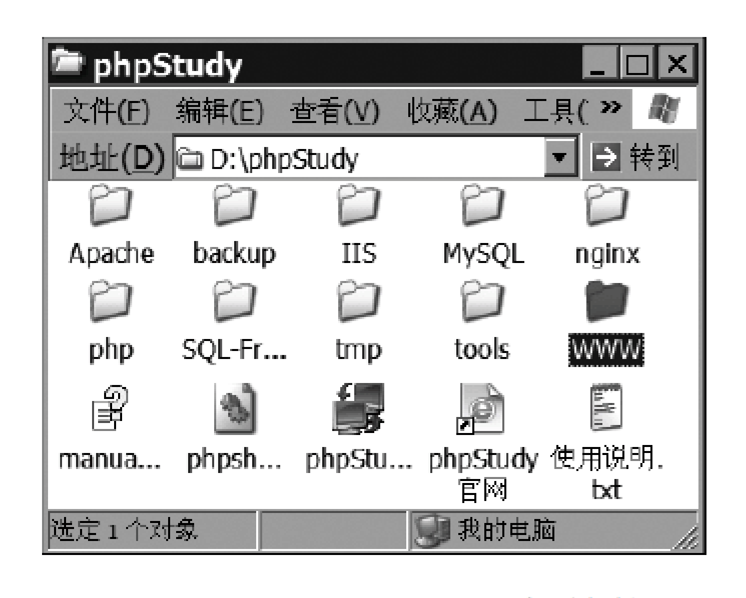


图 1-11　phpStudy 文件结构

夹下或者该文件夹的子文件夹下才能被服务器运行;其他文件夹或者文件也各有用途。

1.5　综合案例——创建第一个 PHP 程序

1. 创建 PHP 程序

本书使用 Dreamweaver CS5 作为 PHP 程序的开发工具。读者也可以使用其他开发工具。最简单的开发工具是记事本。关于 Dreamweaver 的安装读者可以参考其他教科书,本书不做赘述。

【示例】 创建第一个 PHP 程序。

步骤 1 在文件夹 D:\phpStudy\WWW 下面新建文件夹 ch1。启动 Dreamweaver,选择“文件”→“新建”命令,出现如图 1-12 所示的“新建文档”对话框。

步骤 2 在“新建文档”对话框中分别选择“空白页”→PHP→“<无>”。这样便可以创建一个没有布局的 PHP 空白页,如图 1-13 所示。

步骤 3 在如图 1-13 所示中选择“代码”视图,并编写代码(删除原有的全部代码),如图 1-14 所示。

服务器根目录是 D:\phpStudy\WWW,因此所有的 PHP 程序必须保存在文件夹 D:\phpStudy\WWW 中或者其子文件夹中。此处保存(按 Ctrl+S 组合键即可保存)为 D:\phpStudy\WWW\ch1\eg1.php(事先要在 D:\phpStudy\WWW 下建立文件夹 ch1)。



图 1-12　“新建文档”对话框

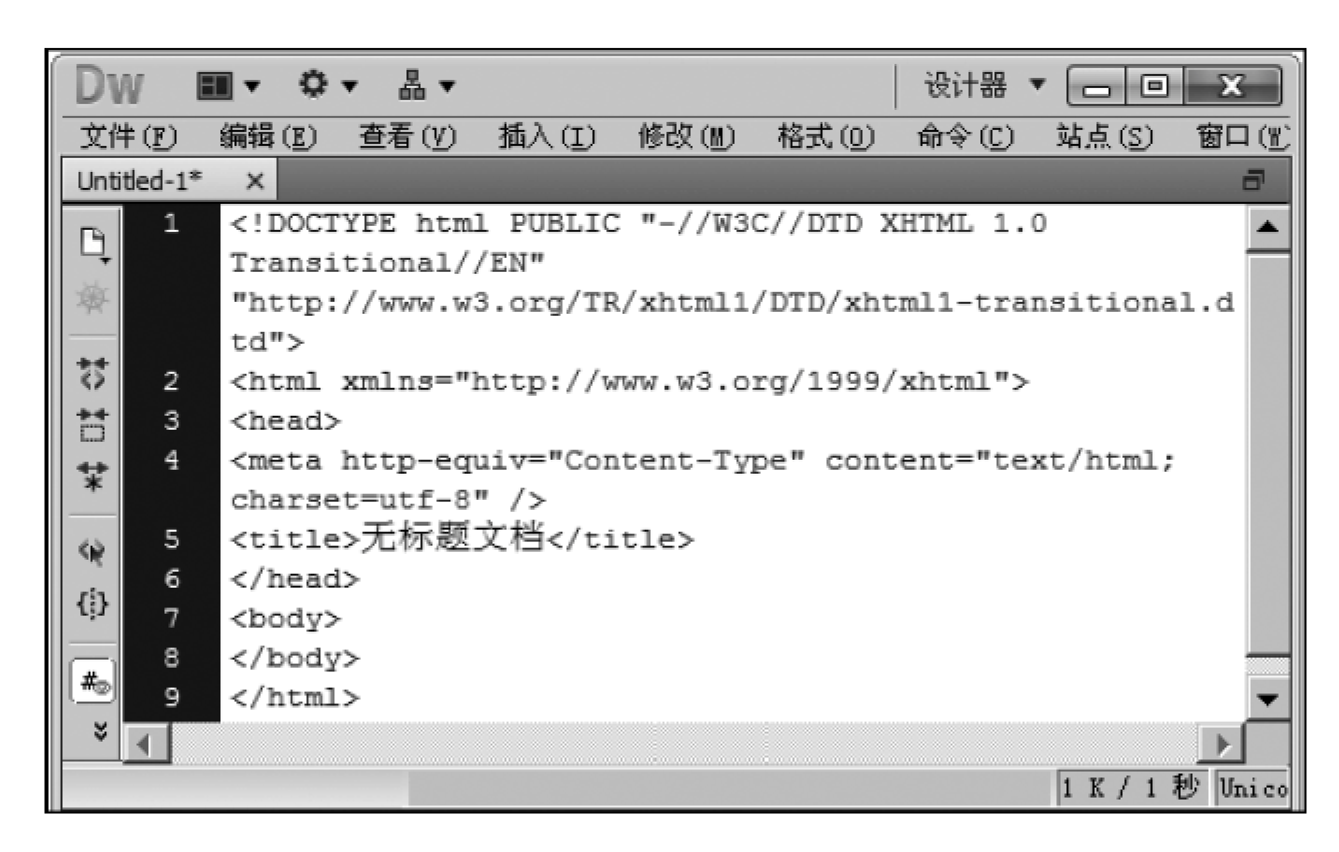


图 1-13　创建空白的 PHP 页面



图 1-14　第一个 PHP 程序

2. 运行 PHP 程序

要运行 PHP 程序,必须先启动 Apache 服务器或者 IIS 服务器(本书使用 Apache 服务器)。如果使用了数据库,则还需要启动 MySQL 数据库服务器。

下面运行 PHP 应用程序。

步骤 1 要想运行 PHP 程序,首先要运行 phpStudy。双击桌面上的 phpStudy 快捷方式图标,出现如图 1-15 所示界面。可以发现在“运行状态”选项区中 Apache 和 MySQL 处于红色状态,表明 Apache 服务器和 MySQL 服务器还没有启动。单击“启动”或者“重启”按钮,启动 Apache 服务器和 MySQL 服务器,此时 Apache 服务器和 MySQL 服务器的状态处于绿色状态,表明 Apache 服务器和 MySQL 服务器都已经启动,见图 1-16。如果 Apache 服务器已经启动,则不需要此步骤。

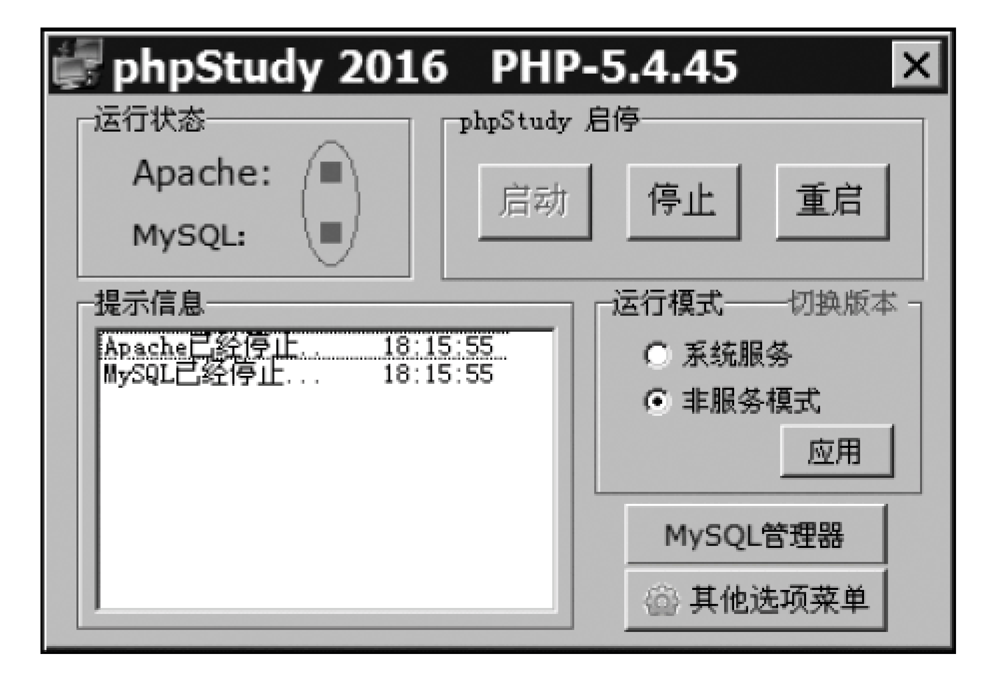


图 1-15 phpStudy 2016 启动界面

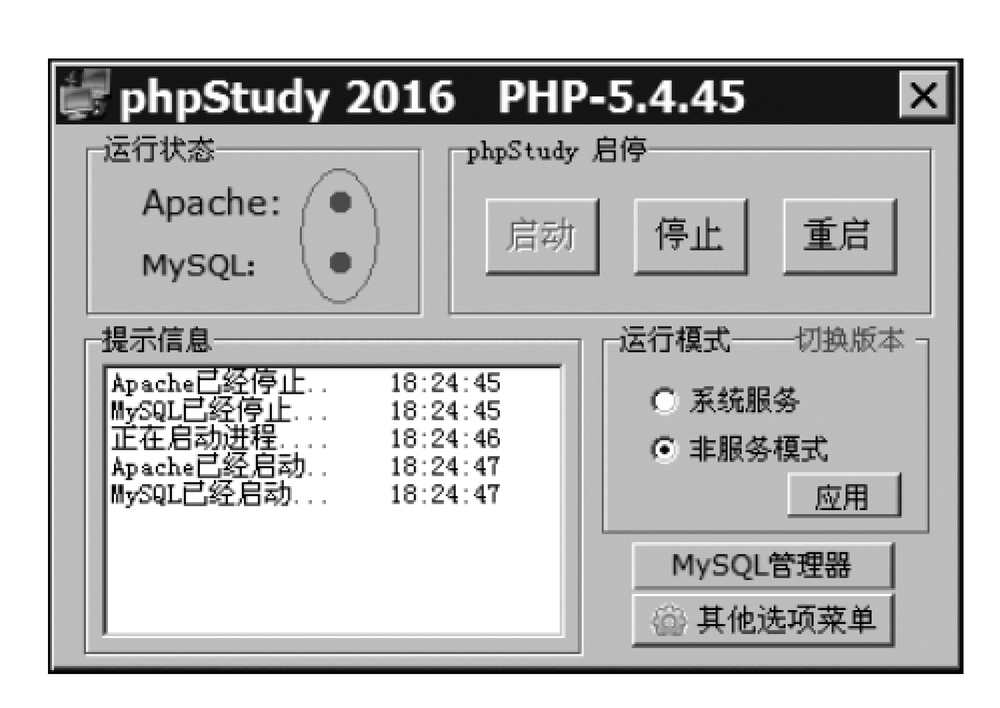


图 1-16 Apache 和 MySQL 服务器已经启动

步骤 2 启动浏览器,在网址中输入 http://localhost/ch1/eg1.php,运行结果如图 1-17 所示,此时程序在浏览器中显示“Hello,World!”。



图 1-17 第一个 PHP 程序的运行结果

1.6 习 题

一、选择题

1. 可以在________网站下载 PHP 工具包。

   A. http://www.php.net　　B. http://www.apache.org

   C. http://www.mysql.com　　D. http://www.asp.net

2. 在 PHP 可用的开发工具中,________是一个强大的网页设计和编辑工具,它的良好设计界面与代码编辑特点受到各级网站建设者的欢迎。

   A. EditPlus　　B. PHPEdit　　C. Dreamweaver　　D. Zend Studio

3. 使用 PHP 开发工具的优点不包括________。

A. 代码中强调不同语句的颜色,使可读性大为改善

B. 支持利用通用的 MVC 框架开发,在 PHP 中可以使用 Zend Framework、CakePHP、CodeLgniter 和 Symfony 等多个应用框架

C. 脚本中包含许多函数和方法,找到函数后直接单击,能够立即打开并定位到该函数所在的文件和相应行,在操作之前可以预览函数和方法的内容

D. 在出现简单的语法错误或者语句丢失时,开发工具应该能够及时提示,开发者可以直接根据提示进行修改

二、上机实习

1. 下载并安装 phpStudy 2016。

2. 编写一个 PHP 应用程序,显示“Hello,PHP!”。

第 2 章　PHP 基本语法

知识点：

- PHP 脚本标记
- PHP 注释
- PHP 的输出

本章导读：

PHP 独特的语法混合了 C、Java 等语言以及 PHP 独创的语法，可以比 CGI 或者 Perl 更快速地执行动态网页。与其他的编程语言相比，PHP 做出的动态页面是将程序嵌入 HTML 文档中去执行，执行效率比完全生成 HTML 要高许多；PHP 还可以执行编译后代码，编译可以实现加密和优化代码运行，使代码运行得更快。本节将介绍 PHP 的入门基础。

2.1　PHP 语法入门

2.1.1　PHP 脚本标记

PHP 文件通常是将 PHP 语句段嵌入 HTML 标记中，其文件内容包括两部分：PHP 标记和 HTML 标记。而 HTML 文件中还可以有其他语言的脚本标记，因此要使用 PHP，需要为该语言添加开始与结束语句的标记，告诉浏览器此处使用的是 PHP 脚本。

PHP 的脚本块以"＜？ php"开始，以"？ ＞"结束。通常可以把 PHP 的脚本块放置在文档中的任何位置。在支持简写的服务器上，还可以使用"＜?"和"？ ＞"来开始和结束脚本块，这种方法称为短标记。为了达到最好的兼容性，本书推荐使用标准形式(＜？ php)，而不是简写形式，如下所示。

```
<?php              ?>
```

其简写形式的格式如下。

```
<?                 ?>
```

除了使用上述两种嵌入方式以外，还有两种嵌入方式，即使用类似 JavaScript 的嵌入方式和使用类似 ASP 的嵌入方式，其标记如下。

```
<script language="php">
…
</script>
```

或

```
<%
…
%>
```

上述 4 种嵌入方法并不是可以直接使用的，在确定浏览器支持上述嵌入方式之后，需要修改 PHP 的配置文件，以确保上述嵌入可以正常使用，如下所示。

- <? php　? >　这种嵌入方法是标准嵌入方法，可直接使用。
- <?　? >　短标记嵌入在默认情况下是不能直接使用的，需要在配置文件中进行修改。找到 php. ini 配置文件，将其 short_open_tag 值修改为 On（默认为 Off），如下所示：

```
short_open_tag = On
```

- <script language="php"></script>　这种嵌入方式是可以直接使用的，不需要修改配置文件。
- <%　%>　这种嵌入方式在默认情况下是不能直接使用的，需要在配置文件中进行修改，将其 asp_tags 值修改为 On（默认为 Off），如下所示：

```
asp_tags =On
```

2.1.2　一个简单的 PHP 程序

下面是一个使用 PHP 脚本标记编写的输出“Hello，world!”的 PHP 程序。

【示例 1】　输出“Hello，world!”。

步骤 1　在文件夹 D:\phpStudy\WWW 下面新建文件夹 ch2。启动 Dreamweaver，选择“文件”→“新建”命令，出现“新建文档”对话框。

步骤 2　在该对话框中分别选择“空白页”→PHP→“<无>”。这样便可以创建一个没有布局的 PHP 空白页。

步骤 3　再选择“代码”视图，并编写代码，按 Ctrl+S 组合键保存代码到 ch2 文件夹，文件名为 eg1. php，如图 2-1 所示。

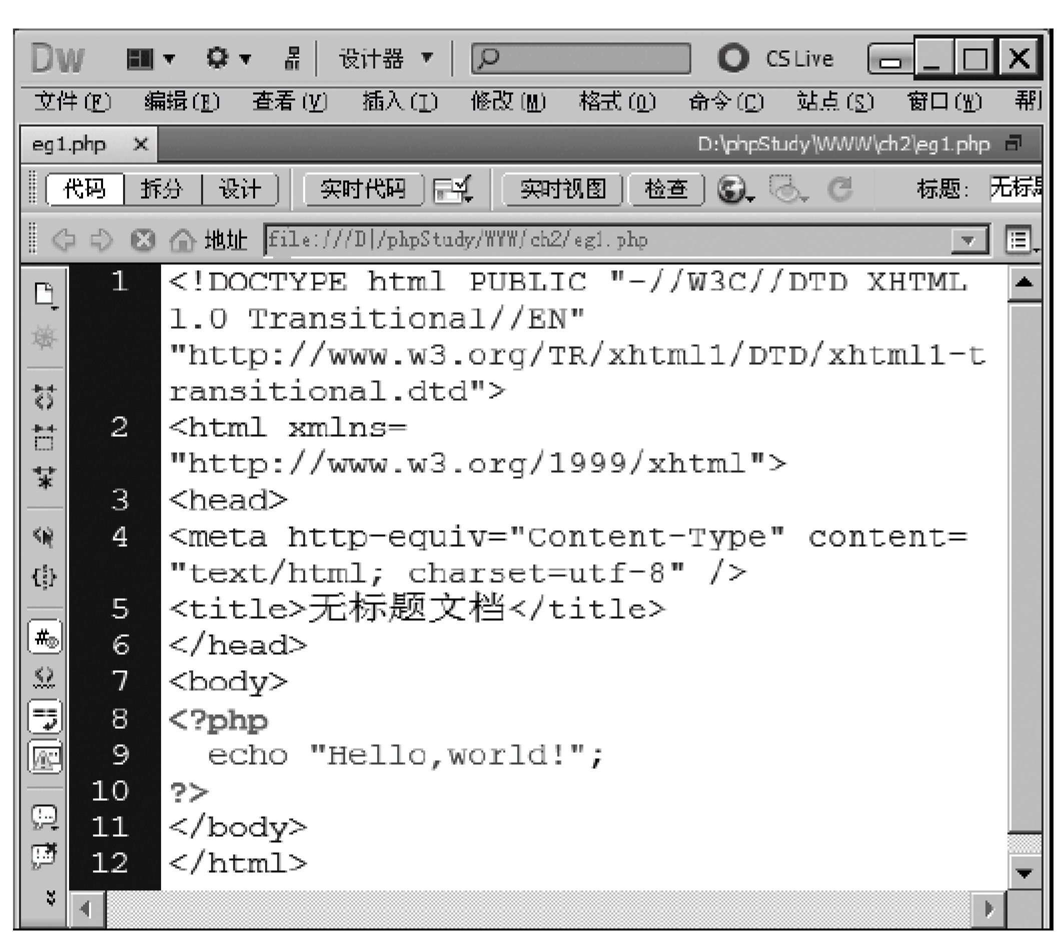


图 2-1　输出“Hello，world!”的代码

从图 2-1 中可以看出 PHP 代码和 HTML 代码是混合在一起的，互相嵌套。运行方法和运行结果和第 1 章的(D:\phpStudy\WWW\ch1\eg1.php)是完全相同的，如图 2-2 所示。



图 2-2　示例 1 的程序运行结果

【示例 2】 输出一个一行两列且宽为 300 的表格。

代码(纯 HTML 代码)可以如图 2-3 所示。

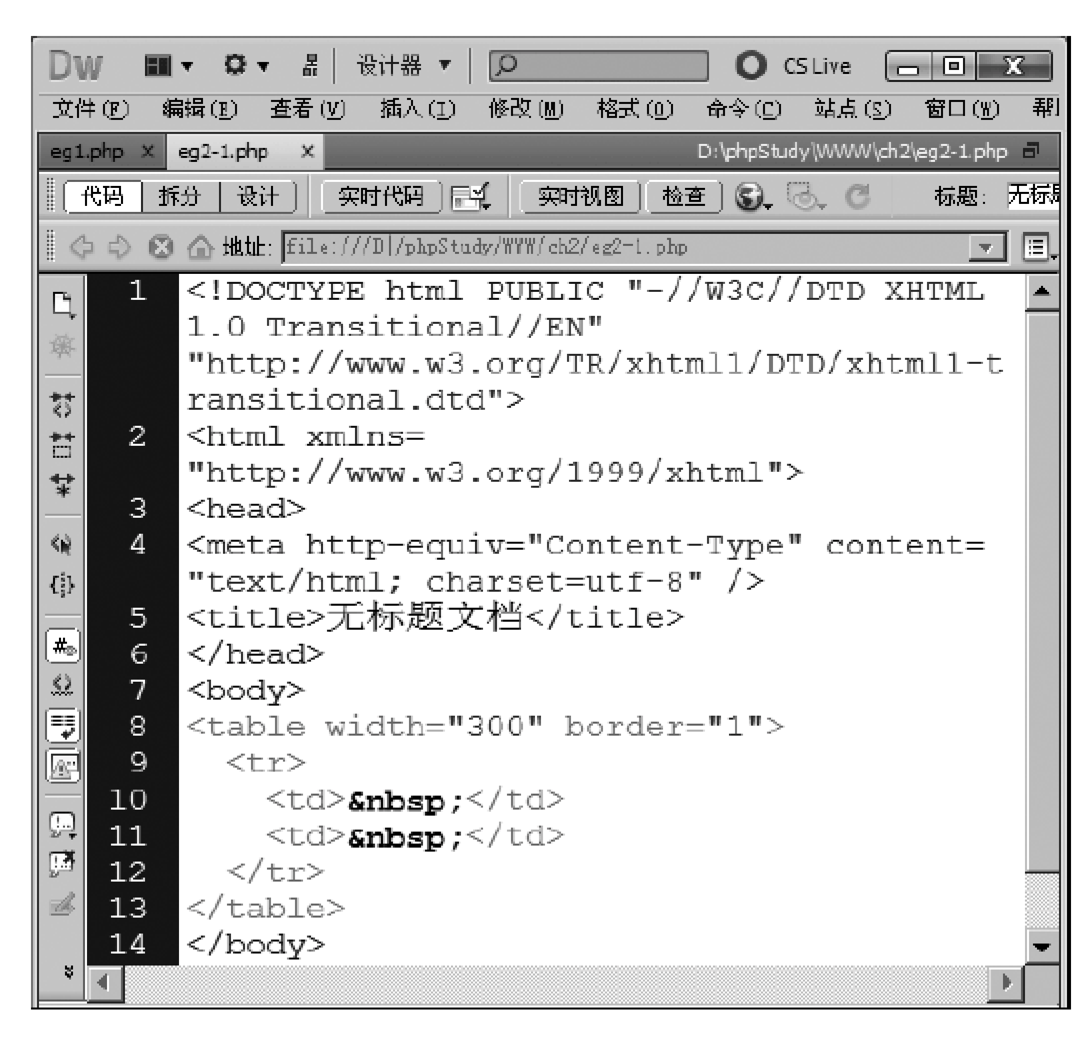


图 2-3　一行两列的表格对应的代码

还可以如图 2-4 所示的代码(HTML 和 PHP 互相嵌套)。

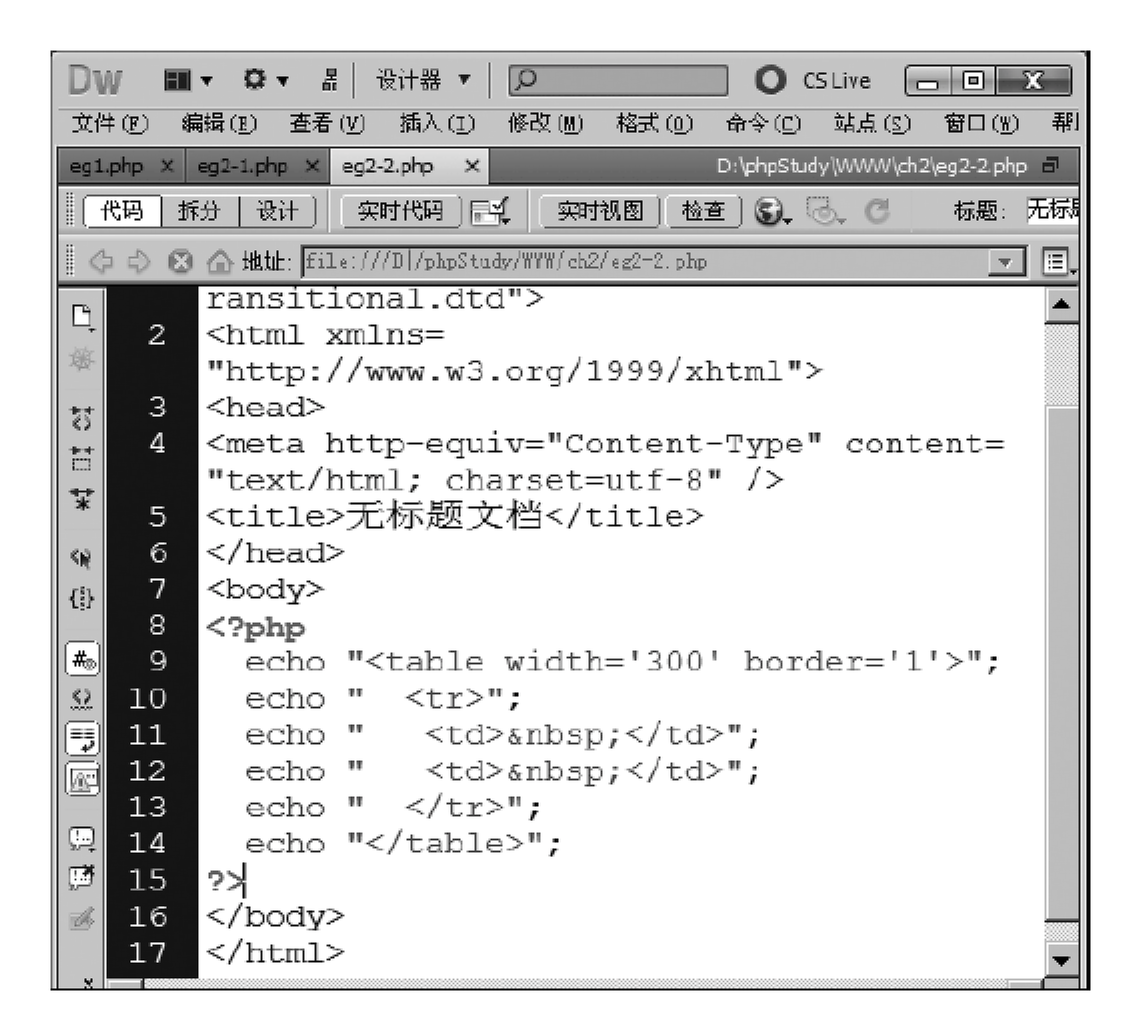


图 2-4　输入一行两列的代码

以上两种代码的运行结果如图 2-5 所示。从图 2-5 中可以看出,不仅 PHP 可以嵌套在 HTML 中,HTML 代码也可以嵌套在 PHP 中。这种互相嵌套的方式,增加了代码书写的灵活性。



图 2-5　输出表格

【示例 3】　用几种方式输出“Hello,world!”。

eg3.php 的代码如图 2-6 所示。

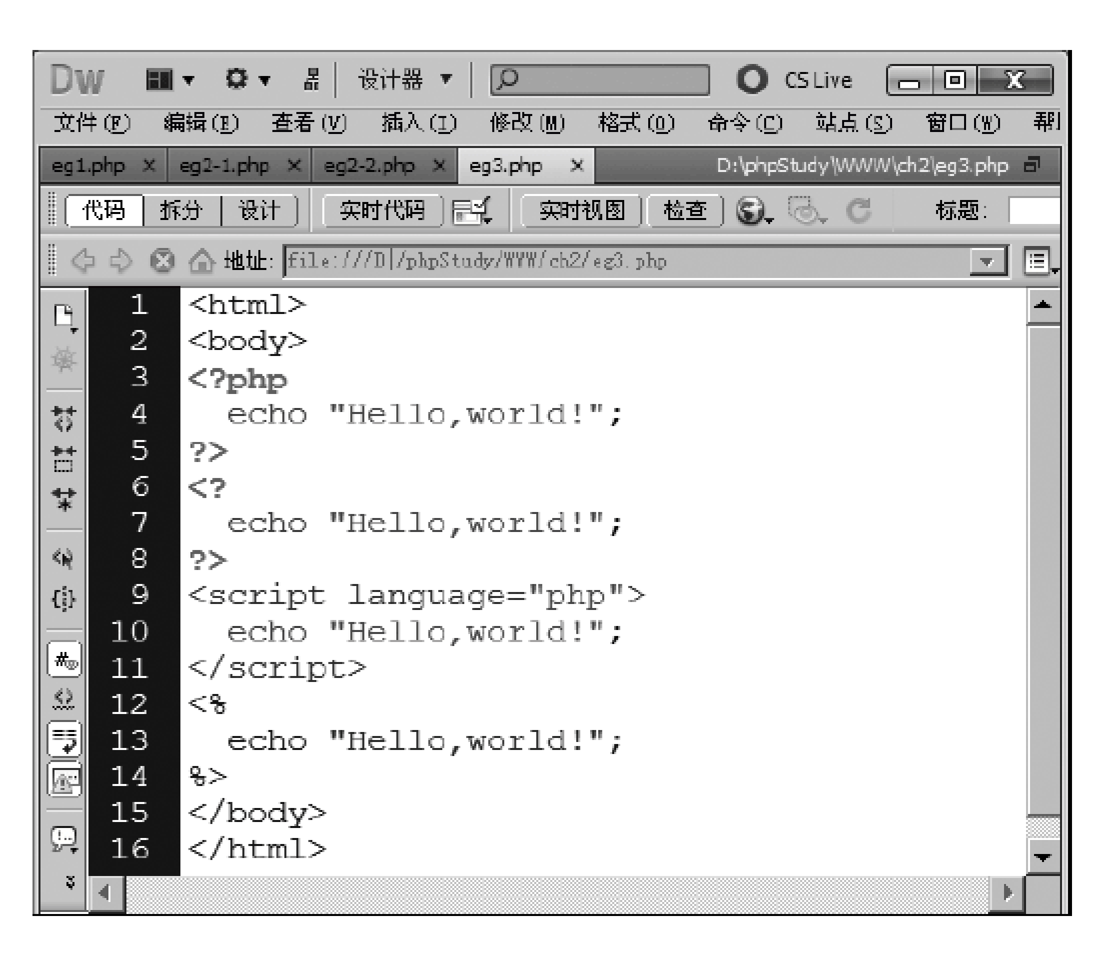


图 2-6　输出“Hello,world!”的代码

eg3.php 的程序运行结果如图 2-7 所示。

我们发现<%和%>没有发生作用,原因是<%和%>的嵌入方式在默认情况下是不能直接使用的,需要在配置文件中进行修改,应将其 asp_tags 值修改为 On(默认为 Off)。找到 php.ini 配置文件,将其 asp_tags = Off 值修改为 On(默认为 Off)。并且需要重新启动 Apache,使设置生效。修改并重启 Apache 后,程序运行结果如图 2-8 所示。



图 2-7　eg3.php 的程序运行结果



图 2-8　修改 php.ini 后的程序运行结果

2.2 PHP 注释和 HTML 注释

几乎任何程序设计语言都有注释语句，PHP 和 HTML 也不例外。注释语句的作用是增加程序的可读性，便于别人看懂，也有利于后期维护。有时候在程序调试过程中可以将不要的语句加上注释，不要急于删除，等确认无误后再删除也不迟。

PHP 的注释分为行注释、块注释。HTML 注释以“<!--”开始，以“-->”结束。千万要注意，不同的注释作用的区域是不一样的。

2.2.1 PHP 行注释

行注释就是单行注释，顾名思义就是只注释代码中的一行。PHP 中的单行注释使用双斜线符号“//”或者井号“#”，放在需要注释的内容的左侧。例如：

```
<?php
  //echo "I love China!";
?>
```

上面程序中的语句“echo "I love China!";”被注释了，因此无法输出字符串“I love China!”。

【示例 4】 输出唐诗《春晓》。

步骤 1 将注释和代码放在同一行。eg4-1 代码如下。

```
<?php
  echo "春晓<br />";                          //输出标题
  echo "春眠不觉晓，处处闻啼鸟。<br />";      //<br />表示换行
  echo "夜来风雨声，花落知多少。<br >" ;      //<br>也表示换行
?>
```

步骤 2 将注释和代码放在不同行。eg4-2 代码如下。

```
<?php
  //输出标题
  echo "春晓<br />";
  echo "春眠不觉晓，处处闻啼鸟。<br />";
  echo "夜来风雨声，花落知多少。<br />";
?>
```

步骤 3 使用#号作为注释符合。eg4-3 代码如下。

```
<?php
  #输出标题
  echo "春晓<br />";
  echo "春眠不觉晓，处处闻啼鸟。<br />";  #<br />用于换行
  echo "夜来风雨声，花落知多少。<br />";
?>
```

上述 eg4-1.php、eg4-2.php 和 eg4-3.php 代码运行结果相同，如图 2-9 所示。



图 2-9　输出唐诗《春晓》

上述程序在输出中文时如果输出乱码，则可以使用语句“header("content-type:text/html;charset=utf-8");”修改页面的编码，使之输出正常的中文字符。还可以使用记事本打开之后另存为编码为 UTF-8 格式的文件，如图 2-10 所示。“保存类型”一定要选择“所有文件”，以免保存为“eg4-3.php.txt”之类的文件名。



图 2-10　使用记事本保存为 UTF-8 格式

2.2.2　PHP 块注释

块注释又称为多行注释，是对多行语句进行的注释，效率比单行注释要高。注释开始的地方使用“/*”，注释结束的地方使用“*/”。

【示例 5】　输出唐诗《春晓》。

eg5-1.php 代码如下。

```
<?php
  /*
  下面的代码输出唐诗《春晓》
  其中 header 用于设置编码格式为 utf-8 格式
  在程序输出乱码时，可以考虑使用 header 改变网页编码
  */
  header("content-type:text/html;charset=utf-8");
  echo "春晓<br />";
  echo "春眠不觉晓，处处闻啼鸟。<br />";
```

```
  echo "夜来风雨声,花落知多少。<br />";
?>
```

上面例子中,在/ * 和 * /中的部分是注释,在程序运行时不执行该部分的内容。

2.2.3　HTML 注释

在 HTML 中使用“<!--”和“-->”进行注释。

【示例 6】 下面的程序不输出唐诗《春晓》。

步骤 1　eg6-1.php 代码如下。

```
<html>
<body>
<!--
<?php
  header("content-type:text/html;charset=utf-8");
  echo "春晓<br />";
  echo "春眠不觉晓,处处闻啼鸟。<br />";
  echo "夜来风雨声,花落知多少。<br />";
?>
-->
</body>
</html>
```

上述代码将不输出唐诗《春晓》,因为整个 PHP 代码都被注释了,注释写在 HTML 里面,PHP 的外面。

步骤 2　eg6-2.php 代码如下。

```
<html>
<body>
/*
<?php
  header("content-type:text/html;charset=utf-8");
  echo "春晓<br />";
  echo "春眠不觉晓,处处闻啼鸟。<br />";
  echo "夜来风雨声,花落知多少。<br />";
?>
*/
</body>
</html>
```

程序运行结果如图 2-11 所示。



图 2-11　没有达到不输出唐诗《春晓》的结果

上面的代码并没有达到不输出唐诗《春晓》的结果，原因在于/＊和＊/注释是 PHP 注释，它只能在 PHP 内部发生作用，也就是在"<? php"之后及"? >"之前发生作用，即(eg6-3. php 代码)：

```
<html>
<body>
<?php
/*
  header("content-type:text/html;charset=utf-8");
  echo "春晓<br />";
  echo "春眠不觉晓，处处闻啼鸟。<br />";
  echo "夜来风雨声，花落知多少。<br />";
 */
?>
</body>
</html>
```

2.3　PHP 的输出

在 PHP 中可以使用 echo 语句来输出，也可以使用 print()函数或者 printf()函数来输出，当然还可以使用 var_dump()函数来输出。

2.3.1　echo 语句

其实 echo 并不是一个函数，它是一个语句结构。因此在使用 echo 时，不一定非要使用括号。echo 的语法格式如下。

```
void echo ( string $arg1 [, string $ ...] )
```

说明：输出 $arg1、$arg2...所有的参数。字符串可以是单引号或者双引号，如果是多个参数就不要使用括号了。例如：

```
<?php
  echo "111","222",'333',"<br />";//字符串可以是单引号或者双引号
  echo ("aa");
  echo ("bb".'cc');              //点号用于连接字符串,"bb".'cc'可以理解为一个参数,即字
                                   符串"bbcc"
  // echo ("aa",'bb','cc');      //此语句错误,有多个参数时不能用括号
?>
```

【示例 7】 输出"我有一头小毛驴，我从来也不骑。"。

eg7. php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");//设置编码格式 utf-8,以免有乱码
  echo "我有一头小毛驴,我从来也不骑。<br />";    //不用括号,用双引号
  echo '我有一头小毛驴,我从来也不骑。<br />';    //不用括号,用单引号
```

```
    echo "我有一头小毛驴,","我从来也不骑。<br />"; //两个参数,不能加括号
    echo ("我有一头小毛驴,我从来也不骑。<br />");  //用括号时只能是一个参数
    echo ("我有一头小毛驴,"."我从来也不骑。<br />");//使用"."号连接时仍然是一个参数
?>
```

【示例 8】 输出一个加法运算。

eg8.php 代码如下。

```
<?php
    $a=12;
    $b=23;
    $c=$a+$b;
    echo '$a+$b=',$c;                              //输出$c 的值 35
?>
```

2.3.2 print()函数

print()函数和 echo 语句的功能差不多,都是输出变量或者字符串。但是 print()是函数,且有 int 类型的返回值。格式如下。

```
int print ( string $arg )
```

输出 arg。print 严格地说也不是一个函数(它是一个语言结构),因此可以不必使用圆括号来括起它的参数列表。

```
<?php
    header("content-type:text/html;charset=utf-8");
    print ("武汉");
    print ("北京");
    print "上海";
    echo print "武汉";                //先输出武汉,再输出 1,因为"print "武汉""语句返回 1
    //print "南京","成都";            //出错,不能输出多个参数
?>
```

print()函数或语句有 int 返回值,一般是 1 或者 0,1 表示输出成功,0 表示输出失败。

```
<?php
    header("content-type:text/html;charset=utf-8");
    echo print "长沙","郑州";
    //print 输出"长沙",再由 echo 输出"print "长沙""的值 1 和"郑州"
?>
```

【示例 9】 输出几个城市。

eg9.php 的代码如下。

```
<?php
    header("content-type:text/html;charset=utf-8");
    //print "南京","成都";            //出错,不能输出几个参数
    echo print "长沙","郑州";
    //print 输出"长沙",再由 echo 输出"print "长沙""的值 1 和"郑州"
    print print "苏州";
```

```
  //先执行“print "苏州"",输出"苏州",再把“print "苏州"”的值 1 输出出来
?>
```

2.3.3　printf()函数

printf()输出格式化字符串。有点类似于 C 语言里面的 printf()函数。printf()函数有返回值。格式如下。

```
int printf ( string $format [, mixed $arg1 [, mixed $arg2 [,…] ]] )
```

依据 format 格式参数产生输出。其中 format 是必选参数，$arg1 是必需的，$arg2、$arg3…是可选的。返回的整数是字符串的长度(对于汉字而言，如果是 utf-8 编码，则一个汉字的长度是 3；如果是 gb2312 编码，则一个汉字的长度是 2)。

格式控制可分为两部分，一部分是用来描述控制格式的符号，另一部分是用来描述数据类型和格式的字符。如“%d”中的“%”用来描述控制格式(起始符号)，而“d”用来描述数据类型和格式(整型格式)。格式控制符及其说明如表 2-1 所示，PHP 中描述数据格式的字符如表 2-2 所示。

表 2-1　格式控制符及其说明

格式控制符	说　明
%	表示格式说明的起始符号，不可缺少
-	有“-”表示左对齐输出，如省略表示右对齐输出
0	有 0 表示指定空位填 0，如省略表示指定空位不填
m.n	m 指域宽，即对应的输出项在输出设备上所占的字符数。n 指精度，用于说明输出的实型数的小数位数。未指定 n 时，隐含的精度为 n=6 位
l 或 h	l 对整型指 long 型，对浮点型指 double 型。h 用于将整型的格式字符修正为 short 型

表 2-2　格式字符及其说明

格式字符	说　明
d 格式	用来输出十进制整数。有以下几种用法。 • %d：按整型数据的实际长度输出。 • %md：m 为指定的输出字段的宽度。如果数据的位数小于 m，则左端补以空格；若大于 m，则按实际位数输出。 • %ld：输出长整型数据
o 格式	以无符号八进制形式输出整数。对长整型可以用%lo 格式输出。同样也可以指定字段宽度，用%mo 格式输出
x 格式	以无符号十六进制形式输出整数。对长整型可以用%lx 格式输出。同样也可以指定字段宽度，用%mx 格式输出
u 格式	以无符号十进制形式输出整数。对长整型可以用%lu 格式输出。同样也可以指定字段宽度，用%mu 格式输出
c 格式	输出一个字符。例如，%c 表示将参数认定为一个整数，显示为对应的 ASCII 字符

续表

格式字符	说　明
s 格式	用来输出一个字符串,有以下几种用法。 • %s:例如,printf(%s,CHINA)输出 CHINA 字符串(不包括双引号)。 • %ms:输出的字符串占 m 列,如字符串本身长度大于 m,则突破 m 的限制,将字符串全部输出;若字符串长度小于 m,则左补空格。 • %-ms:如果字符串长度小于 m,则在 m 列范围内,字符串向左靠,右补空格。 • %m.ns:输出占 m 列,但只取字符串左端 n 个字符。这 n 个字符输出在 m 列的右侧,左补空格。 • %-m.ns:其中 m、n 含义同上,n 个字符输出在 m 列范围的左侧,右补空格。如果 n>m,则自动取 n 值,即保证 n 个字符正常输出
f 格式	用来输出实数(包括单、双精度),以小数形式输出。有以下几种用法。 • %f:不指定宽度,整数部分全部输出并输出 6 位小数。 • %m.nf:输出共占 m 列,其中有 n 位小数,如数值宽度小于 m,左端补空格。%-m.nf:输出共占 m 列,其中有 n 位小数,如数值宽度小于 n,右端补空格
e 格式	以指数形式输出实数。可用以下形式。 • %e:数字部分(又称尾数)输出 6 位小数,指数部分占 5 位或 4 位。 • %m.ne 和%-m.ne:m、n 和"-"字符的含义与前相同。此处 n 指数据的数字部分的小数位数,m 表示整个输出数据所占的宽度
g 格式	自动选 f 格式或 e 格式中较短的一种输出,且不输出无意义的零

【示例 10】 使用 printf()函数输出。

eg10.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $a=printf("钢笔%d 元",6);
  echo "<br />$a<br />";
  $b=printf("圆珠笔%3.2f 元,本子%d 元",4.326,5);
  echo "<br />$b";
?>
```

输出结果如图 2-12 所示。



图 2-12　printf()函数输出结果

说明:

- header 语句设置编码为 utf-8 格式,在这种格式下,一个汉字的长度是 3。
- "$a=printf("钢笔%d 元",6);"语句中,%d 使用 6 来代替,%d 表示整型数。
- "$b=printf("圆珠笔%3.2f 元,本子%d 元",4.326,5);"语句中,%3.2f 使用 4.326 代替,四舍五入为 4.33,%f 表示实数;一个函数中可以有多个参数,这里是两

个参数。

- 双引号是可以解析出字符串中的变量的，因此在字符串中如果有变量，则输出的是变量的值。单引号则不能解析字符串中的变量。

【示例 11】 使用 printf()函数输出。

eg11.php 代码如下。

```
<?php
  $num1 =123456789;
  $num2 =-123456789;
  $char =50;                                //ASCII 字符 50 是 2
  //注释:格式值 "%%" 返回百分号
  printf("%%b =%b <br>",$num1);             //二进制数
  printf("%%c =%c <br>",$char);             //ASCII 字符
  printf("%%d =%d <br>",$num1);             //带符号的十进制数
  printf("%%d =%d <br>",$num2);             //带符号的十进制数
  printf("%%e =%e <br>",$num1);             //科学计数法(小写)
  printf("%%E =%E <br>",$num1);             //科学计数法(大写)
  printf("%%u =%u <br>",$num1);             //不带符号的十进制数(正)
  printf("%%u =%u <br>",$num2);             //不带符号的十进制数(负)
  printf("%%f =%f <br>",$num1);             //浮点数(视本地设置)
  printf("%%F =%F <br>",$num1);             //浮点数(不视本地设置)
  printf("%%g =%g <br>",$num1);             //短于%e 和 %f
  printf("%%G =%G <br>",$num1);             //短于%E 和 %f
  printf("%%o =%o <br>",$num1);             //八进制数
  printf("%%s =%s <br>",$num1);             //字符串
  printf("%%x =%x <br>",$num1);             //十六进制数(小写)
  printf("%%X =%X <br>",$num1);             //十六进制数(大写)
  printf("%%+d =%+d <br>",$num1);           //符号说明符(正)
  printf("%%+d =%+d <br>",$num2);           //符号说明符(负)
?>
```

程序运行结果如图 2-13 所示。各个语句的解释见语句后面的注释。

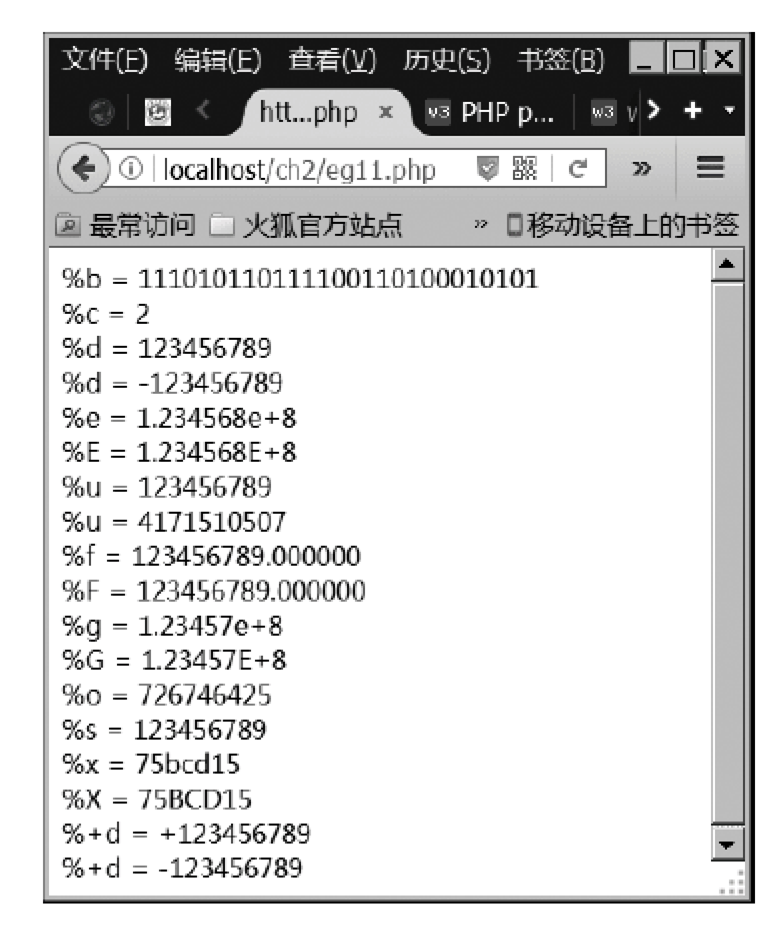


图 2-13　应用 printf()函数

2.3.4 var_dump()函数

var_dump()函数用于参数的相关信息。有时候使用 echo 语句并不能得到输出参数的详细信息，达不到程序想要的效果。此时可以考虑使用 var_dump()函数，它能得到变量的详细信息，这样有利于代码的书写。var_dump()函数的格式如下。

```
void var_dump ( mixed $expression1 [, mixed $expression2... ] )
```

该函数没有返回值，$expression1 是必需的，$expression2... $expression*n* 是可选的。该函数的作用是输出这些表达式的值和类型。

```
<?php
  $a=true;
  $b=false;
  echo $a;      //输出 1
  echo $b;      //没有输出
?>
```

要想了解 $b 的信息，此时需要用到 var_dump。代码如下。

```
<?php
  $a=true;
  $b=false;
  var_dump($a,$b);
?>
```

在进行程序设计时，有时候需要先使用 var_dump()函数试探出表达式的详细信息，再来制订后面的代码方案。

【示例 12】 判断子字符串。

判断子字符串相关知识点虽然在后面章节，但是此处读者可以着重考察 var_dump()函数的作用。

步骤 1 eg12-1.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $mystring = 'abc';
  $findme   = 'x';
  $pos = strpos($mystring, $findme);
  if ($pos >=0) {
    echo "是子字符串";
  } else {
    echo "不是子字符串";
  }
?>
```

上述代码的运行结果居然显示“是子字符串”，这个结果显然是错误的。如果我们仔细研究(使用 var_dump) $pos 的值，发现该值有时候是 false(不是子字符串)，有时候是整型值 n(表示子字符串的位置，从 0 开始)，于是问题来了，在 PHP 中 0==false 是成立的，而且 false>=0 也是成立的，因此不能使用 $pos>=0 来判断是否是子字符串。

步骤 2　eg12-2.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $mystring = 'abc';
  $findme   = 'x';
  $pos = strpos($mystring, $findme);
  var_dump($pos);  //在得知$pos详细信息后(包括数据类型和值),该语句可以去掉
  if ($pos === false) {
    echo "不是子字符串";
  } else {
    echo "是子字符串";
  }
?>
```

在 PHP 中,运算符“===”表示数据类型和值都相等。上述程序经运行,结果正确。

2.4　综合案例——职工个人信息的输出

使用本章所学知识,输出职工个人信息。需要输出的文本如下。

```
东方仪表厂职工信息如下。
职工号    姓名      职称|职务              基本工资(元)
001      刘芳      董事长                12345.00
002      诸葛靓    教授级高级工程师        8989.45
003      司马胜男  高级工程师              8000.12
004      晁有文    工程师                  5600.00
005      袁野      副厂长                  9020.50
```

要求对上述信息进行格式化输出,使各种信息对齐输出,更加美观。

eg13.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  printf("%54s","东方仪表厂职工信息如下。<br />");
  print("职工号    姓名      职称|职务              基本工资(元)<br />");
  print("001      刘芳      董事长                12345.00<br />");
  print("002      诸葛靓    教授级高级工程师        8989.45<br />");
  print("003      司马胜男  高级工程师              8000.12<br />");
  print("004      晁有文    工程师                  5600.00<br />");
  print("005      袁野      副厂长                  9020.50");
?>
```

2.5　习　　题

一、填空题

1. PHP 的标准嵌入方式,其开始标记为________。

2. PHP 的嵌入方式有________种。
3. PHP 单行注释可使用________或“//”。
4. PHP 的输出函数有________ print()、printf()与 sprintf()。
5. PHP 多行注释的开始和结束标记为________和“*/”。

二、选择题

1. 以下不属于 PHP 标准嵌入方式的是________。
   A. <?php?>
   B. <??>
   C. <script language="php"></script>
   D. <?---->
2. 下列函数无返回值的是________。
   A. echo()　　B. print()　　C. printf()　　D. sprintf()
3. 下列函数返回字符串类型返回值的是________。
   A. echo()　　B. print()　　C. printf()　　D. sprintf()
4. 下列不属于在 PHP 标记内对 PHP 语句进行注释的是________。
   A. //　　B. #　　C. <!-- -->　　D. /* */
5. 下面关于格式控制的说法错误的是________。
   A. %表示格式说明的起始符号
   B. 有“-”表示右对齐输出，如果省略表示左对齐输出
   C. 有 0 表示指定空位填 0，如果省略表示指定空位不填
   D. l 对整型指 long 型，对实型指 double 型。h 用于将整型的格式字符修正为 short 型
6. 下面关于格式的说法错误的是________。
   A. d 用来输出十进制整数
   B. o 以无符号八进制形式输出整数
   C. x 以无符号十六进制形式输出整数
   D. u 以有符号十进制形式输出整数

三、程序设计题

1. 商品折扣信息输出。

商店的促销活动经常有对商品的折扣处理。结合本章内容，处理一家商店的商品折扣信息。该商店现有需要打折的商品，如下所示。

毛呢大衣原价 488 元，打 88 折。
羊绒衫原价 458 元，打 7 折。
围巾原价 100 元，打 65 折。
毛衣链原价 30 元，打 5 折。

输出上述打折商品的信息，要求包含商品名称、原价、折扣和现价。其中原价为 3 位整数；现价为整数，根据其数值本身的长度来输出。在编写代码的同时，合理使用注释。

2. 考生成绩输出。

对考生成绩的统计是任课教师每年必不可少的工作。结合本章内容，处理一个班级学

生的考试成绩。该班级学生的成绩信息如下。

张飞数学考试成绩 68。
张飞语文考试成绩 77。
张飞英语考试成绩 66。
关羽数学考试成绩 86。
关羽语文考试成绩 79。
关羽英语考试成绩 67。

输出上述两位学生的考试平均分，要求如下。

（1）分别输出两位学生各自的平均分，要求以整数形式显示。

（2）输出该班级的语文平均分，要求保留两位小数。

（3）输出该班级的数学平均分，要求保留两位小数。

第 3 章　PHP 数据处理

知识点：

- 常量和变量的定义以及变量的作用域
- 数据类型(标准数据类型和复合数据类型)
- 运算符以及运算符的优先级

本章导读：

本章将讲述 PHP 的常量和变量的定义和作用域、各种数据类型和运算符。这些内容是学习 PHP 的基础。

3.1　标准数据类型

PHP 中的标准数据类型包含单个值,通常被称为单一数据类型。常用的标准数据类型有布尔型(boolean)、整型(integer)、浮点型(float 和 double)以及字符串型(string)。

3.1.1　布尔型

在 PHP 中布尔型用于表示事物的真假,比如“婚否”,是否“党员”。布尔值用 true 和 false 来表示。已婚可以使用 true 来表示,未婚可以使用 false 来表示。

【示例 1】 输出刘备的政治面貌和婚姻状态。

步骤 1　eg1-1.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $marriaged=false;
  $party=true;
  echo '刘备的婚姻状态是:'.$marriaged.'<br />';
  echo '刘备的政治面貌是:'.$party.'<br />';
?>
```

程序运行结果如图 3-1 所示。我们发现婚姻状态 false 没有显示。在 PHP 中 true 显示为 1,false 显示为 0。要想显示 true 和 false,可以使用 var_dump()函数。还可以使用强制类型转换为整型,用 1 表示 true,0 表示 false。



图 3-1　eg1-1 的程序运行结果

步骤 2　eg1-2.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $marriaged=false;
  $party=true;
  echo '刘备的婚姻状态是:'.(int)$marriaged.'<br />';
  echo '刘备的政治面貌是:'.(int)$party.'<br />';
?>
```

程序运行结果如图 3-2 所示。



图 3-2　将布尔型转换为整型再输出

3.1.2　整型

整型数就是不包含小数部分的数，包含十进制、十六进制、八进制数，整型数包括正数、负数和 0。十六进制数前面要加上 0x，八进制数前面要加 0。

【示例 2】　输出一些整型数。

eg2.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $int1=0123;
  $int2=0x1ab;
  $int3=123;
  $int4=1234567890;
  $int5=12345678901;
  echo $int1.'<br />';
  echo $int2.'<br />';
  //双引号可以解析引号中的变量，字符串中没有变量时尽量用单引号，代码更优
  echo "$int3<br />";
  var_dump($int4);
  echo '<br />';
  var_dump($int5);
?>
```

程序运行结果如图 3-3 所示。



图 3-3　输出整型数

整型数表示的范围是有限度的，如果超过这个限度，则自动转化为浮点型。本例中 1234567890 是整型数，而 12345678901 则超过整型数表示的范围而自动转换为浮点型。整型数表示范围与平台有关，无须死记硬背。

3.1.3 浮点型

浮点型就是实型(real)，包括单精度类型(float)和双精度类型(double)。浮点型常用于表示货币、重量、距离等，在整型无法表示的情况下用浮点型。可以使用科学计数法来表示浮点型。

【示例 3】 输出一些浮点型数。

eg3.php 代码如下。

```
<?php
  $f1=1.23;
  $f2=1.0;            //带小数点，是浮点型
  $f3=12345678901;    //超过整型数表示的范围，是浮点型
  $f4=1e3;            //表示 1000，用科学计数法表示，是浮点型
  //$f5=1e4.0;        //e 后面不能带小数点，即使小数点后面是 0 也不能带，非法
  $f6=1.23e-2;        //e 后面可以是负数
  var_dump($f2);      //浮点型
  echo '<br />';
  var_dump($f3);      //浮点型
  echo '<br />';
  var_dump($f4);      //浮点型
  echo '<br />';
  var_dump($f6);      //浮点型
?>
```

3.1.4 字符串型

字符串是一个字符序列，可以看成数组。PHP 中可以表示的字符包含 256 种，也就是说使用一个字节来表示一个字符。可以使用单引号或者双引号来表示字符串，字符串中的特殊字符可以使用转义的方法来表示。

1. 单引号

单引号可以用来表示一个字符串。当字符串中仅仅包含单纯的字符，而不包含变量或者转义字符时，使用单引号比较好，代码更加优化。当然使用双引号也是可以的。

【示例 4】 单引号用法。

eg4.php 代码如下。

```
<?php
  $str1='Hello,\nworld!';  //\n 表示两个普通的字符，不表示换行
  $a='Wuhan';
  $b='I like $a!';          //$a 表示两个普通字符，不表示 Wuhan
  echo nl2br($str1);        //nl2br 能把\n 输出为换行
  echo '<br />';
  echo $b;
?>
```

程序运行结果如图 3-4 所示。

2. 双引号

双引号可以解析出字符串中的特殊字符以及变量，普通字符不做任何改变。使用双引号可以使代码非常灵活。

【示例 5】 双引号用法。

```
<?php
  $str1="Hello,\nworld!";   //\n 表示换行
  $a='Wuhan';
  $b="I like $a!";          //$a 表示 Wuhan
  echo nl2br($str1);
  echo '<br />';
  echo $b;
?>
```

程序运行结果如图 3-5 所示。



图 3-4　单引号的程序运行结果



图 3-5　双引号的程序运行结果

【示例 6】 使用大括号来分开字符串中的变量。

eg6.php 代码如下。

```
<?php
  $a='Wuhan';
  $ab='Jingzhou';
  $x="abc$ab$a";            //abcJingzhouWuhan
  $y="abc{$a}b$a";          //abcWuhanbWuhan
  echo $x.'<br />';
  echo $y
?>
```

3. 转义字符

PHP 可以识别多种特殊字符，例如转义字符，包括换行符等。表 3-1 列出了一些常用的转义字符。

表 3-1　PHP 能够识别的几种转义字符

转义序列	说　明
\n	换行符
\r	回车符
\t	水平制表符
\\	反斜线

续表

转义序列	说　　明
\$	美元符号
\"	双引号
\[0-7]{1,3}	在正则表达式中用于表示八进制
\x[0-9A-Fa-f][1,2]	在正则表达式中用于表示十六进制

【示例 7】 巧用单双引号以及转义字符的输出。

eg7. php 代码如下。

```
<?php
  $a="I'm a teacher\n";                    //因字符串中带单引号,所以使用双引号
  $b='I said:"You are a good student!"';     //因字符串中带双引号,所以使用单引号
  echo $a.'<br />';
  echo $b.'<br />';
  $c="I have a book,\nthe name is Sanguoyanyi.";
  $d="a\\bc\m\tcd\r";
  echo $c.'<br />';
  echo $d.'<br />';
?>
```

说明:

- 字符串"a\\bc\m\tcd\r"前面两个斜线在程序中显示一个斜线。第一个斜线告诉程序后面是特殊的转义字符。
- 因为\m 不是特殊的转义字符,所以该斜线是普通字符。
- \t 表示水平制表。
- \r 表示回车。

【示例 8】 在字符串中如果想要使用 html 的格式标签,可以使用＜article＞和＜/article＞。则在＜article＞和＜/article＞之间的 HTML 标签都可以解释出来。在开发考试系统时,可以用＜article＞和＜/article＞来定制格式,将格式保存在字符串和数据库中。

eg8. php 代码如下。

```
<?php
  $a="
<article>
  <h1>Internet Explorer 9</h1>
  <p>Windows Internet Explorer 9</p>
  <font size='+1' color='#FF0000'>Wuhan</font>
</article>
  ";
  echo $a;
?>
```

程序运行结果如图 3-6 所示。



图 3-6　字符串中使用＜article＞和＜/article＞

3.1.5　复合数据类型

复合数据类型将多个具有相同类型的项聚集起来，表示为一个实体，主要包括数组、对象、资源数据类型和空类型。下面简单介绍数组类型和对象类型，详细介绍见后面的章节。

1. 数组

数组就是把具有相同数据类型的项集合在一起进行处理，并按特定的方式进行排列和引用。每个数组元素都有两个要素，即值(value)和键(key)。可以通过数组元素的键(key)访问到数组元素的值(value)。数组元素的键又叫作索引或者下标，默认值是从 0 开始的序列，索引也可以是字符类型。

例如：

```
$a[0]=1;
$a[1]='01';
$a[2]='刘备';
$a[3]='male';
$a[4]=true;
```

$a 是一个一维数组，它的键分别是 0、1、2、3、4，还可以写成：

```
$a[]=1;
$a[]='01';
$a[]='刘备';
$a[]='male';
$a[]=true;
```

此时 $a 的键自动从 0 开始，并能自动加 1。还可以使用 array 函数创建数组，可以写成：

```
$a=array(1,'01','刘备','male',true);
```

数组元素的键可以是数字和字符，例如：

```
$a=array(0=>1,'no'=>'01','name'=>'刘备','sex'=>'male',true);
```

$a 数组的键分别是 0、'no'、'name'、'sex' 和 1；最后一个元素 true 的键是 1，即前面的 0 自动加 1。每一个数组元素的表示方法为：键=>值。

数组元素也可以表示为：

```
$a[0]=1;
$a['no']='01';
$a['name']='刘备';
$a['sex']='male';
$a[1]=true;     //键如果不写，也会默认为 1，即前面的 0 再加 1 所得
```

【示例 9】 使用不同的方法定义数组。

eg9-1.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
```

```
    //$a1 的键分别是 0、1、2、3
    $a1[]='01';
    $a1[]='宋江';
    $a1[]='男';
    $a1[]=90;
    //$a2 的键分别是 0、'no'、'sex'、'age'
    $a2[0]=76;
    $a2['no']='201701';
    $a2['sex']='female';
    $a2['age']=18;
    //$a3 的键分别是 0、1、'age'、2
    $a3=array('liubei',90,'age'=>23,true);
    //$a4 的键分别是'no'、'sex'和 0
    $a4=array('no'=>1,'sex'=>'male',90);
    var_dump($a1,$a2,$a3,$a4);
?>
```

程序运行结果如图 3-7 所示。



图 3-7　用不同的方法定义数组

下面介绍二维数组。

所谓二维数组，既是数组的元素又是数组，因此可以使用两个键（即索引）来访问数组元素。

例如：

```
<?php
    $a[0]=98;
    $a[]=99;
    //$b 是二维数组，因为$b 的元素中有数组类型
    $b=array(12,90,$a,'sex'=>'male');
    var_dump($b);
    echo '<br />';
    echo $b[0].'<br />';                    //输出 12
    echo $b[2][1];                          //输出 99
?>
```

【示例 10】　定义二维数组。

步骤 1　eg10-1.php 代码如下。

```
<?php
    //$stu 是二维数组
    $stu[0]=array('01', 'liubei',  'male',  90);
```

```
  $stu[1]=array('02', 'guanyu');
  $stu[2]=array('03', 'diaochan','female',78);
  $stu[3]=array('04');
  var_dump($stu);
?>
```

步骤 2　eg10-2.php 代码如下。

```
<?php
  //$stu 是二维数组
  $stu=array(array('01', 'liubei',  'male', 90),
             array('02', 'guanyu'),
             array('03', 'diaochan','female',78),
             array('04'));
  var_dump($stu);
?>
```

需要说明的是,二维数组的各个元素并不一定都是数组,元素的个数也不一定是相等的。元素值的类型也不一定是相同的。只是在现实中"规则的"二维数组更容易处理。

关于数组的知识,将在后面更加详细地介绍。

2. 对象

PHP 支持的另一种复合数据类型是对象,面向对象编程的一个核心概念就是对象。在创建对象之前必须先定义一个类。所谓类,就是对象的概括、对象的模型,是一个宏观的概念。一般来说一个类中定义了属性、方法和其他成员。把属性、方法和其他成员整合在一起的过程就是封装。对象就是类的实例,即实例化的类。有关类、对象、属性、方法、封装等知识将在面向对象编程的章节中详细讲解。

下面介绍如何进行类的定义。

【示例 11】　定义 Student 类,并使用该类创建对象。

eg11.php 代码如下。

```
<?php
  class Student{              //定义类
    //包含两个属性:$stu_name 和$stu_score
    private $stu_name='Liubei';
    public $stu_score;
    //包括一个方法 show()
    function show(){
      echo $this->stu_name.'<br />';
      echo $this->stu_score.'<br />';
    }
  }
  $st1=new Student();         //使用 new 创建一个对象$st1
  $st1->stu_score=90;         //访问属性
  $st1->show();               //调用方法
?>
```

说明:

- 使用 class 来定义类,一对大括号之间就是类的定义部分。

- 类的成员包含属性和方法。
- 使用 new 来创建类的对象。
- 使用对象名->方法(或属性)来访问类的成员(属性或方法)。

程序运行结果如图 3-8 所示。



图 3-8 对象编程

3.2 数据类型转换

PHP 中包含多种数据类型。表达式的数据类型是可以通过类型转换来改变的。具体分为两类:强制数据类型转换和自动数据类型转换。

3.2.1 强制数据类型转换

强制数据类型转换是指将一个表达式转换为与原来类型不同的另外一种数据类型。强制数据类型转换需要在代码中明确地申明需要转换的数据类型,一般情况下,强制数据类型转换可以将范围大的数据类型转换为取值范围小的数据类型。PHP 数据类型强制转换一般可以使用 3 种方式,下面分别进行介绍。

1. 在要转换的表达式之前加上用括号括起来的目标数据类型

这种方式是非常常见的一种数据类型转换方式,在表达式之前插入如表 3-2 所示的操作符,即可实现数据类型的强制转换。

表 3-2 数据类型的强制转换操作符

数据类型转换操作符	转换为(数据类型)	数据类型转换操作符	转换为(数据类型)
(int)或(integer)	整型	(array)	数组类型
(float)、(double)或(real)	浮点型	(object)	对象类型
(string)	字符串型	(bool)或(boolean)	布尔型

【示例 12】 数据类型转换。

eg12. php 代码如下。

```
<?php
  $old=-12.6-3;
  $new1=(int)$old;          //转换变量为整型,
  $new2=(int)($old+4);      //转换表达式为整型
  var_dump($new1,$new2);    //int(-15)和 int(-11)会去尾,不会四舍五入
  $i1=12;
  $f1=(float)$i1;
```

```
    echo '<br />';
    var_dump($f1);                //float(12)
    $f2=12.3;
    $a=array($f2);                //转换为数组类型
    echo '<br />';
    var_dump($a);                 //array(1) { [0]=>float(12.3) }
    $f3=1.23;
    $st1=(string)($f3+12);        //将表达式转换为字符串类型
    echo '<br />';
    var_dump($st1);               //string(5) "13.23"
    //将$f3转换为字符串类型,然后在和 12 相加时又转换为数值类型,强制类型转换优先级高于加
      法运算
    $f4=(string)$f3+12;
    echo '<br />';
    var_dump($f4);                //float(13.23)
    $b1=(boolean)$i1;             //非 0 转换为 true
    $b2=(bool)($i1-12);           //0 转换为 false
    $n=null;
    $b3=(bool)$n;
    echo '<br />';
    var_dump($b1,$b2,$b3);        //bool(true)、bool(false)、bool(false)
    $o=(object)$i1;
    echo '<br />';
    var_dump($o);                 //object(stdClass)#1 (1) { ["scalar"]=>int(12) }
    $a=1.656;
?>
```

程序运行结果如图 3-9 所示。

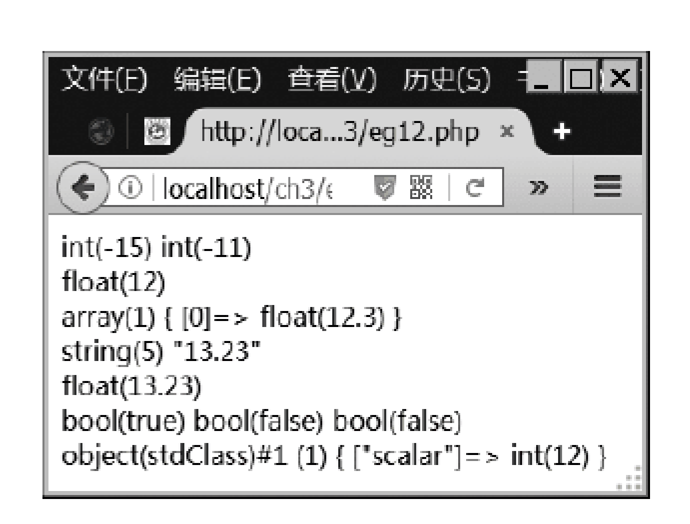


图 3-9　数据类型强制转换

说明：

- 使用这种强制数据类型转换，可以对变量、表达式、函数进行。
- “$f4=(string)$f3+12;”先将$f3 强制转换为字符串类型，再进行加法运算。进行加法运算时会自动转换为数值类型，结果是数值类型。
- 在转换为布尔类型时，空字符串、没有元素的数组、0、NULL(NULL 是空类型)转换为 false，非 0 转换为 true。
- 一般情况下各种数据类型都可以转换为 object 类型，因为 object 是所有类的父类，转换的结果是变量成为对象的一个属性，属性名是 scalar。
- 实型转换为整型数时实现去尾取整，不论正负都一样，不会四舍五入。

2. 使用具体类型的转换函数

使用具体类型的转换函数也可以实现数据类型的强制转换,常用的数据类型转换函数有 4 个,分别是 intval()、floatval()、doubleval()和 strval()。其中 intval()函数用于将表达式强制转换为 int 类型;floatval()函数用于将表达式强制地转换为 float 数据类型;doubleval()函数是 floatval()函数的别名;strval()函数用于将表达式强制地转换为 string 数据类型。4 个函数的格式分别如下。

(1) int intval(mixed $var [, int $base = 10]):通过使用指定的进制 $base 转换(默认是十进制),返回变量 $var 的 integer 数值。intval() 不能用于 object,否则会产生 E_NOTICE 错误并返回 1。

(2) float floatval(mixed $var):返回变量 $var 的 float 数值。$var 可以是任何标量类型。不能将 floatval() 用于数组或对象。

(3) doubleval()函数:它是 floatval()函数的别名,格式完全相同。

(4) string strval(mixed $var):返回 $var 的 string 值。参见 string 文档获取更多关于字符串转换的信息。var 可以是任何标量类型。不能将 strval()函数用于数组或对象。

【示例 13】 使用函数进行强制数据类型转换。

eg13.php 代码如下。

```
<?php
  $str="1234.567abc";
  var_dump(intval($str));          //int(1234)表示十进制
  var_dump(intval($str,8));        //int(668)表示八进制的 1234 变成十进制为 668
  var_dump(doubleval($str));       //float(1234.567)
  var_dump(floatval($str));        //float(1234.567)
  var_dump(strval($str));          //string(11)为"1234.567abc"
  $f=1234.5678;
  var_dump(strval($f));            //string(9)为"1234.5678"
  $b=false;                        //看成数值型 0 或者空字符串
  var_dump(intval($b));            //int(0)
  var_dump(doubleval($b));         //float(0)
  var_dump(floatval($b));          //float(0)
  var_dump(strval($b));            //string(0),即""
  $c=true;                         //看成数值型 1 或者字符串"1"
  var_dump(intval($c));            //int(1)
  var_dump(doubleval($c));         //float(1)
  var_dump(floatval($c));          //float(1)
  var_dump(strval($c));            //string(1),即"1"
  $mynumstr ="100,000,000.75";
  $num =doubleval(str_replace(",","",$mynumstr));  //先用空字符替换逗号再转换为实型
  var_dump($num);                  //float(100000000.75)
?>
```

3. 使用 settype()函数转换数据类型

在 PHP 中 settype()函数用于设置变量的数据类型。格式为:

```
bool settype ( mixed &$var, string $type )
```

该函数用于将变量 var 的类型设置成 type。$var 是要转换的变量,type 可能的值为:

- "boolean"（或为"bool"，从 PHP 4.2.0 起开始用）
- "integer"（或为"int"，从 PHP 4.2.0 起开始用）
- "float"（只在 PHP 4.2.0 之后可以使用，旧版本中使用的是"double"，现已停用）
- "string"
- "array"
- "object"
- "null"（从 PHP 4.2.0 起开始用）

成功时返回 true，失败时返回 false。

【示例 14】 使用 settype()函数转换变量的数据类型。

eg14.php 代码如下。

```
<?php
  $foo="5bar";                    //string
  $bar=true;                      //boolean
  settype($foo,"integer");        //$foo 现在是 5(integer)
  settype($bar,"string");         //$bar 现在是"1" (string)
  var_dump($foo,$bar);
  $b1=settype($foo,"integer");    //转换成功,$b1 为 true
  $b2=settype($bar,"string");     //转换成功,$b2 为 true
  var_dump($b1,$b2);              //即 bool(true)、bool(true)
  //"$b3=settype(3+2.6,"integer");"表达错误,settype 的第 1 个参数必须是变量
  $var="true";
  settype($var,'bool');
  var_dump($var);                 //true
  $var="false";
  settype($var,'bool');
  var_dump($var);                 //true
  $var="";                        //只有空字符串转换为布尔型时才转换为 false
  settype($var,'bool');
  var_dump($var);                 //false
?>
```

3.2.2　自动数据类型转换

PHP 是弱类型语言，它对于类型的定义比较松散，它会根据实际情况进行自动数据类型转换，无须强制进行数据类型转换。

【示例 15】 数据类型自动转换。

eg15.php 代码如下。

```
<?php
  $f1="1";
  $f1++;                          //因为要做自加运算,所以$f 自动转换为数值型 1
  var_dump($f1);
  $f2='';
  $f2=$f2+1;                      //$f2 自动转换为数值型 0,然后数值类型再做加法运算
  var_dump($f2);                  //int(1)
  $f3=1;
```

```
    $f3=$f3."2";            //1 自动转化为字符串"1",然后字符串再做连接运算
    var_dump($f3);          //string(2) "12"
    $a="12";
    $b=34;
    $c=$a+$b;               //因为是做加法运算,所以将$a 自动转换为数值型 12,再做加法运算
    $d=$a.$b;               //因为是做字符串连接运算,所以将$b 字符转换为字符串"34",再做
                              字符串连接运算
    var_dump($c,$d);        //int(46) 和 string(4) "1234"
    $x=array();
    $y=$x && true;          //空数组自动转换为布尔型 false,再进行逻辑运算
    var_dump($y);           //bool(false)
    $c=true;
    $d=2;
    $e=$c+$d;               //int(3),把$c 自动转换为整型数 1,再做加法运算
    $f=$c.$d;               //string(2) "12",把$c 转换为字符串"1",再做连接字符串运算
    var_dump($e,$f);        //int(3) 和 string(2) "12"
?>
```

说明:

- 自动类型转换一般要根据运算符来决定到底怎样转换类型。
- “$f1="1"; $f1++;”因为要做加法运算,所以要将字符串 $f1 自动转换为数值类型 1。
- “$a="12"; $b=34; $c= $a+ $b;”因为要做加法运算,所以要将字符串 $a 自动转换为数值型 12,再做加法运算。
- “$a="12"; $b=34; $d= $a. $b;”因为要做字符串连接运算,所以要将 $b 自动转换为字符串类型"34",再做字符串连接运算。
- “$x=array(); $y= $x && true;”因为要做逻辑运算,所以把空数组转换为 false,再做逻辑与运算。
- “$c=true; $d=2; $e= $c+ $d;”因为要做加法运算,所以将 $c 自动转换为整型数 1,再做加法运算,得到整数型 3。
- “$c=true; $d=2; $f= $c. $d;”因为要做字符串连接运算,所以将 $c 自动转换为字符串"1",再做字符串连接运算,得到 string(2) "12"。

3.2.3 数据类型函数

除了 intval()函数、float()函数、strval()函数和 settype()函数之外,PHP 还提供了一些与数据类型相关的函数。

1. 使用 gettype()函数获得表达式数据类型

gettype()函数的格式如下。

```
string gettype ( mixed $var )
```

返回表达式的数据类型,这些数据类型可能是:

- "boolean"(从 PHP 4 起);
- "integer";
- "double"(由于历史原因,如果是 float 则返回"double",而不是"float");

- "string";
- "array";
- "object";
- "resource"(从 PHP 4 起);
- "NULL"(从 PHP 4 起);
- "unknown type"。

【示例 16】 使用 gettype()函数获取表达式的数据类型。

eg16.php 代码如下。

```
<?php
  $a1="true";
  var_dump(gettype($a1));            //string
  var_dump(gettype(3+2.2));          //double
  $a2=array(1,'name'=>'liubei');
  var_dump(gettype($a2));            //array
  var_dump(gettype($a3));            //NULL
  $a4=(object)($a1);
  var_dump(gettype($a4));            //object
  $a5=true;
  var_dump(gettype($a5));            //boolean
?>
```

2. 使用函数来检测表达式是否是某种类型

PHP 中提供了一类函数用于检测表达式是否是某种数据类型,这些函数包括 is_int()、is_integer()、is_double()、is_float()、is_string()、is_array()、is_object()、is_bool()、is_null()、is_numberic()、is_resource()和 is_scalar()。这些函数的作用相似,都是用来判断表达式是否是某种数据类型,因此可以将它们归为一类。下面以 is_int()函数为例来统一讲解。

is_int()函数的格式为:

```
bool is_int ( mixed $var )
```

该函数判断表达式是否是整型,如果表达式是整型则返回 true,不是整型则返回 false。is_integer()函数是该函数的别名。其他函数用法类似。

【示例 17】 使用 is_int()等函数进行表达式数据类型的判断。

eg17.php 代码如下。

```
<?php
  $a ='15';
  var_dump(is_int($a));              //false
  $a +=0;
  var_dump(is_int($a));              //true
  $b=false;
  var_dump(is_bool($b));             //true
  $c=array(1,'name'=>'liubei','sex'=>'male');
  var_dump(is_array($c));            //true
  $d=12;
  var_dump(is_numeric($d));          //true
```

```
  $e="It's a string.";
  var_dump(is_string($e));            //true
  var_dump(is_null($f));              //true
  var_dump(is_object($a));            //false
  var_dump(is_object((object)$a));    //true
  var_dump(is_float(4.5));            //true
  var_dump(is_float(4));              //false
?>
```

3.3 变　　量

变量就是指数据存储单元，可以用来存储数据，在程序中变量的值可以发生改变，因此被称为变量。PHP中的变量的数据类型是比较弱的，在程序中不仅变量的值可以发生改变，就连变量的类型也可以发生改变。

3.3.1　变量的声明

变量是指在程序运行的过程中其值可以随时会发生改变的量。变量名则是用来标识变量的，这样便于程序访问。在PHP中规定变量名是以美元符号$打头，然后是PHP标识符。PHP变量名遵循以下一些原则。

- PHP变量名之前必须有一个美元符号$。一般不把$符号看成变量名的一部分。
- PHP变量名长度必须小于或者等于255个字符，这些字符可以是大小写英文字母、数字和下画线。
- 变量名不能以数字开头。
- 变量名大小写敏感，即区分大小写，但是建议使用小写字符。
- 变量名不要和函数同名，以免混淆。
- 不建议使用汉字字符作为变量名。
- 建议变量名做到见名知义，比如$name、$stu_name可以表示姓名或者学生姓名。

【示例18】　下面声明了一些变量，有合法的也有非法的。

eg18.php代码如下。

```
<?php
  $name;        //合法
  $Name;        //合法,但不建议用
  $_sex;        //合法
  $stu_sex;     //合法
  $afgdkk;      //合法,但不建议用
  $3name        //非法的
  $name1;       //合法
  $name2;       //合法
  $姓名;        //合法,但不建议用
?>
```

说明：下面是一些好习惯。

- 变量名不宜太长。
- 变量名要见名知义。
- 变量名用小写。

3.3.2　变量的赋值

所谓变量的赋值，就是通过赋值语句给变量值。PHP 中的赋值号是“＝”，在 PHP 中赋值有两种方式：值赋值和引用赋值。

1. 值赋值

值赋值就是直接给变量赋予值，即把表达式的值分配给变量，在程序运行的过程中变量的值是保存在计算机内存单元中的。变量的值是可以随时发生改变的，甚至连类型也可以发生改变。

【示例 19】 下面是一些赋值语句。

eg19.php 代码如下。

```
<?php
  $a='Beijing';            //赋值为 Beijing,字符串类型
  $b=70;                   //赋值为 70,数值类型
  $c=false;                //赋值为 false,布尔类型
  $a='I love '.$a;         //重新对$a 赋值,$a 值变为'I love Beijing'
  $d=$b+30;                //将表达式$b+30(即 100)赋值给$d
  $c="It's a string."      //$c 重新赋值,类型也发生改变,是字符串类型
  $stu_name="$a";          //双引号可以解析字符串中的变量,变量值是'I love Beijing'
  $x='$b';                 //$x 为字符串类型,值为'$b',单引号不会解析$b 为 70
?>
```

从上述代码中可以看出，变量的值和类型在程序运行过程中都是可以变化的。在 PHP 中可以把常量、变量或表达式赋值给变量。

2. 引用赋值

所谓引用赋值，就是把一个变量的地址应用到另一个变量，即多个变量对应内存中的同一个内存地址。也可以理解为同一变量具有不同的变量名。

在 PHP 中，实现引用赋值需要使用符号 &，实际上在有些语言中 & 是表示内存地址的，这就不难理解为什么要使用 & 符号来实现引用赋值了。

先看一个例子：

```
<?php
  header("content-type:text/html;charset=utf-8");
  $x=1;
  $y=$x;                          //值赋值
  $x=2;
  echo '$x 值为'.$x.'<br />';       //输出:$x 值为 2
  echo '$y 值为'.$y.'<br />';       //输出:$y 值为 1
?>
```

运行该程序，发现当 $x 的值从 1 变成 2 时，$y 并没有发生变化，仍然是 1。因为 $x 和 $y 是两个不同的变量，在内存中是两个不同的内存单元，它们互相独立，互不影响。

再看下面的引用赋值例子。

【示例 20】 下面是引用赋值，请读者观察 $ x 值和 $ y 值的变化。

eg20. php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $x=1;
  $y=&$x;                    //引用赋值
  $x=2;
  echo '$x 值为'.$x.'<br />';      //输出：$x 值为 2
  echo '$y 值为'.$y.'<br />';      //输出：$y 值为 2
?>
```

从运行结果可以看出：对 $ y 进行引用赋值后(即 $ y 指向了 $ x 的内存地址)，当 $ x 的值从 1 变成 2 之后，我们发现 $ y 的值也变成了 2，原因是 $ y 和 $ x 其实就是一个变量。

3.3.3 动态变量

动态变量又称为可变变量，变量的名称保存在另外一个变量中，也可以理解为动态变量以另外一个变量的值作为自己的变量名。动态变量可以为程序带来灵活性。

例如：

```
<?php
  $x='hello';
  $$x='world';
  echo $$x.'<br />';          //输出为 world
  echo ${$x}.'<br />';        //输出为 world
  echo $hello;                //输出为 world
?>
```

从上述程序的运行结果可以看出：$ x 值为'hello'，该值又是动态变量 $ $ x 的变量名，即变量 $ $ x 表示变量 $ hello，二者值都是'world'。还可以把 $ $ x 写成 $ { $ x}，这样 PHP 解释器会先把大括号中的 $ x 解析为'hello'，再来把 $ { $ x}解析为 $ hello。

【示例 21】 下面是动态变量的例子，请注意 $ $ country 的值。

eg21. php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $china='Zhongguo';
  $japan='Riben';
  $russia='Eluosi';
  $country='china';             //$$country 是可变变量，其变量名是$country 的值
  echo $$country.'<br />';      //将$$country 理解为$china，输出为 Zhongguo
  $country='japan';
  echo $$country.'<br />';      //将$$country 理解为$japan，输出为 Riben
  $country='russia';
  echo ${$country};             //将${$country}理解为$russia，输出为 Eluosi
?>
```

3.3.4　变量的作用域

变量的作用域是指变量的有效范围。变量的有效范围与变量声明的位置有关系。在 PHP 中变量的作用域有 4 种情况：局部变量、全局变量、函数参数和静态变量。

1. 局部变量

在函数里面(包括在类中定义的方法)声明的变量是局部变量，顾名思义，局部变量只能在局部范围内使用，在 PHP 中局部变量只能在声明这个变量的函数内部有效。如果在函数外部再来赋值或者访问，则系统认为是赋值或者访问了另一个同名的不同变量。局部变量的使用可以消除程序模块间的副作用。

例如：

```
<?php
  function fun1(){
    $a=2;                          //$a 是局部变量，在函数 fun1()之外是无效的
  }
  $a=1;
  fun1();
  echo $a;                         //输出 1，此$a 并不是函数中声明的那个局部变量$a
?>
```

2. 全局变量

在函数外面声明的变量就是全局变量，全局变量在整个 PHP 程序(包括函数内部)中都是有效的。但是，在函数内部默认的变量都是局部变量。

例如：

```
<?php
  $name = "Liubei";
  function changeName(){
    $name = "Guanyu";              //函数内部声明的变量默认是局部变量
  }
  changeName();
  echo "My name is " . $name . "<br/>";  //输出:My name is Liubei
?>
```

上面程序的运行结果是输出“My name is Liubei”。为什么会是这样呢？原来在 PHP 中，在函数内部使用的变量都会默认为局部变量，因此 changeName()函数并不能改变全局变量 $name 的值。

如果要在函数内部使用全局变量，则需要使用 global 关键字。global 的用法为：

```
global 变量名 1[,变量名 2...];
```

虽然 global 也可以使用在函数外部，但是没有任何意义，因为函数外部声明的变量肯定是全局变量。

【示例 22】　修改上面的例子，注意 $name 的变化。

eg22.php 代码如下。

```
<?php
```

```
  global $name;                //本语句可以不要,因为在函数外部声明的变量肯定是全局变量
  $name ="Liubei";
  function changeName(){
    global $name,$no;          //告诉系统:函数内部使用的$name 和$no 是全局变量
    $name ="Guanyu";
    $no=2;
    $sex='male';               //$sex 没有使用 global 声明过,它是局部变量
  }
  changeName();                //修改全局变量$name 为'Guanyu'
  echo "My name is " . $name . "<br/>";     //输出 My name is Guanyu
  var_dump($sex);              //$sex 是 NULL 值,不是'male'
?>
```

程序运行结果是输出 My name is Guanyu。因为在 changeName()函数中赋值的是全局变量 $ name,即整个 PHP 程序中的 $ name 变量是一个变量。

3. 函数参数

在 PHP 中有些函数是带参数的,这些参数的值来自调用语句,函数调用结束参数的值不复存在。此处仅做简单介绍,更复杂的介绍见自定义函数等章节。

【示例 23】 函数参数举例。

eg23. php 代码如下。

```
<?php
  function fun1($arg1,&$arg2){          //$arg1 和$arg2 是形式参数
     $arg1=$arg1+1;
     $arg2=$arg2+100;
     echo '$arg1='.$arg1.'<br />';      //输出为$arg1=2
     echo '$arg2='.$arg2.'<br />';      //输出为$arg2=200
  }
  $a=1;
  $b=100;
  fun1($a+0,$b);                        //$a+0 和$b 是实际参数
  echo '$a='.$a.'<br />';               //输出为$a=1
  echo '$b='.$b;                        //输出为$b=200
?>
```

该例中 $ arg1 和 $ arg2 都是函数参数,它们的值是从调用语句按位置顺序传递过去的,第一个参数 $ arg1 接受 $ a+0 的值,而第二个参数 $ arg2 则接受 $ b 的值。 $ arg1 对应的实际参数可以是表达式(如本例中的 $ a+0);而 $ arg2 前面冠以 & 符号,表示该参数是“引用传递”,这样的参数只能接受变量(此处为 $ b,绝不可以写成 $ b * 1 等表达式形式)。函数调用完毕,则 $ arg1 和 $ arg2 不复存在。由于 $ arg2 是引用传递,也就是说在调用函数时,$ arg2 和实际参数 $ b 指向的是同一个存储单元,因此改变 $ arg2 的值就是改变 $ b 的值,所以最后输出的 $ b 值为 200。

4. 静态变量

在 PHP 中使用 static 关键字来声明静态变量。静态变量与函数的形式参数不同,函数调用完毕,其形式参数变量不复存在,而静态变量在函数调用完毕后仍然会保存在内存中(其值会保存下来),下一次调用时该值仍然会发生作用。

【示例 24】　静态变量举例。

```
<?php
  header('content-type:text/html;charset=utf-8');
  function visit(){
    static $count=0;
    $count++;
    echo "第{$count}次调用 visit()函数!<br />";
    //若$count 在字符串最后,则不需要加{}分隔后面的内容
  }
  visit();  //输出:第 1 次调用 visit()函数!
  visit();  //输出:第 2 次调用 visit()函数!
  visit();  //输出:第 3 次调用 visit()函数!
?>
```

在上面程序中,第 1 次调用 visit()函数并执行 static $count=0 语句时,系统检测 $count 为静态变量,为该变量分配存储单元,且在调用完毕后 $count 仍然保存在内存中(值为 1),因此第 2 次调用 visit()函数是在执行语句 $count++时,在原值(值为 1)基础上加 1 的,所以第 2 次调用结束后 $count=2;同理,第 3 次调用结束后 $count=3。

3.3.5　变量的销毁

在 PHP 中提供了一种变量自动销毁的机制:即在一个函数调用结束后,函数内部的局部变量自动销毁;一个 PHP 程序运行结束,其中的变量(全局变量)也会自动销毁。有时候为了提高程序性能,也会人工销毁一些变量。比如在使用巨大的多维数组时会占据较多的存储单元,这样不利于程序快速运行,因此会把后面不再使用的变量提前销毁。

销毁变量的方法有两种:一种是赋值为 NULL;另一种是使用函数 unset(变量列表)。unset()函数的格式为:

```
void unset ( mixed $var [, mixed $ ...] )
```

该函数用于销毁一个或多个变量,至少要有一个变量。

【示例 25】　变量的销毁。

eg25.php 代码如下。

```
<?php
  $a=12;
  $b='I love China!';
  $c=true;
  $e=56;
  for($i=0;$i<=100000;$i++)
    $d[]=$i;
  //$d是一个比较大的数组,在程序不用时应将它销毁
  unset($a,$b);      //销毁$a,$b
  $c=NULL;           //销毁$c
  unset($b,$d);      //销毁$b、$d。虽然$b已被销毁,重复写并无语法错误
  var_dump($a,$b,$c,$d);
?>
```

上述代码中使用 unset 将 $a、$b 和 $d 销毁，使用赋值 NULL 将 $c 销毁。注意，$d 是一个非常大的数组，销毁有利于内存空间的回收利用，改善了程序性能。用 var_dump($a, $b, $c, $d)语句检测到 $a、$b、$c 和 $d 四个变量都不存在了。$e 在整个 eg25.php 程序运行结束后也会自动销毁。

3.4 常　　量

与变量一样，常量也是 PHP 中非常重要的概念。但是和变量不同的是，PHP 中的常量在程序运行的过程中不会改变，也不允许改变，也就是说不允许将常量置于赋值号的左边。

3.4.1 常量的定义

像 1、false、true、'hello，world'、2.34 等数据，它们的值在程序运行的过程中是不会改变的，它们是常量。而 abs(-9.6)、1+0、'hello，'. 'PHP'、3.4+6 则是表达式，因为它们带有运算符或函数。

1. 使用 define()函数定义常量

在 PHP 中有一种常量，我们常常使用符合来表示它，本节讨论的就是这种类型的常量。常量的定义方法为：

```
bool define ( string $name, mixed $value [, bool $case_insensitive = false ] )
```

说明：

- 函数返回值是 boolean 类型，设置成功时返回 true，或者在失败时返回 false。
- name 是常量名，符合 PHP 标识符命名规则，如果大小写不敏感，则建议使用小写。
- value 是常量的值，仅允许标量和 null。标量的类型是 integer、float、string 或者 boolean。也能够定义常量值的类型为 resource，但并不推荐这么做，可能会导致未知状况的发生。
- case_insensitive 用于设置大小写是否敏感，如果设置为 true，该常量则大小写不敏感。默认是大小写敏感的(值为 false)。比如，CONSTANT 和 Constant 代表了不同的值。

【示例 26】 常量的定义。

eg26.php 代码如下。

```
<?php
  define("PI1",3.14);                 //默认是区分大小写的
  define("PI2",3.1416,false);         //第 3个参数为 false时表示区分大小写
  define("PI3",3.1415926,true);       //第 3个参数为 true时表示不分大小写
  define("pi",3.1415926,true);        //不分大小写，建议小写
  define("_pi",3.14);                 //合法
  define("3PI",3.14);                 //非法
  echo 'PI1='.PI1.'<br />';           //3.14
  echo 'PI2='.PI2.'<br />';           //3.1416
  echo 'PI3='.PI3.'<br />';           //3.1415926
```

```
  echo 'PI1='.pi1.'<br />';        //不存在 pi1
  echo 'PI2='.pi2.'<br />';        //不存在 pi2
  echo 'PI3='.pi3.'<br />';        //存在 pi3
  echo _pi;                        //3.14,也可以把_pi 写成 constant ("_pi")
?>
```

【示例 27】 常量的定义。

eg27. php 代码如下。

```
<?php
  define("PI1",3.14);              //默认是区分大小写的
  define("PI2",3.1415926,true);    //第 3 个参数为 true 时表示不分大小写。建议用小写
  define("PI3",pi2);               //可以使用常量来定义常量。pi2 是已经定义过的常量
  define("PI1",3.1416);     //PI1 已定义,产生一个 notice(非错误),但是 PI1 仍然为 3.14
  PI1=3.14;                        //错误,不允许给常量赋值
  $a=PI1*3*3;
  echo PI1.'<br />';               //仍然是 3.14
  echo PI3.'<br />';
  function testconstant(){
     echo PI1.'<br />';            //常量是全局的
     define("ALPHA",PI1/2);        //常量是全局的,在此处定义,但在函数外可用
  }
  testconstant();                  //输出 3.14
  echo ALPHA;                      //输出 1.57,也可以把 ALPHA 写成 constant ("ALPHA")
?>
```

常量定义非常简单,但是在定义常量时仍需注意以下几点。

- 常量的前面不带 $ 符号;
- 可以使用常量来定义常量;
- 不能重新定义已经定义过的常量;
- 不允许给常量赋值;
- 在引用常量时不能加引号,双引号是无法解析常量的,这点与变量是不一样的;
- 常量是全局的,函数外面定义的常量在函数内部可以使用,反之亦然。

2. 使用 defined()函数检查某个名称的常量是否存在

为了避免重复定义常量,可以在定义常量之前先判断要定义的常量是否存在。defined()函数就是用来检查某个名称的常量是否存在的。defined()函数的格式为:

```
bool defined (string $name)
```

说明:

- 该函数返回值为布尔型。若 $name 是常数则返回 true,否则返回 false。
- 如果要检查一个变量是否存在,请使用 isset()。defined()函数仅对 constants 有效。如果要检测一个函数是否存在,使用 function_exists()。

【示例 28】 常量的检测。

eg28. php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
```

```
  …
  if (!defined('pi'))
     define('pi',3.1415926,true);
  $radius=3.0;
  $area=pi*$radius*$radius;
  echo "圆的面积为:$area";        //输出为"圆的面积为:28.2743334"
?>
```

上述代码用于求圆的面积。先判断常量 pi 是否存在,若不存在则定义该常量(值为 3.1415926)。如果存在(在前面的代码中已经定义过了)则无须重复定义,直接使用该常量来求圆的面积即可。

3.4.2　类的常量

在面向对象的程序设计中,有时候需要为类定义变量,这时候使用 define()函数是错误的。PHP 提供了一个名叫 const 的关键字用于定义类的常量。类的常量在定义之后是不能人为地修改的。类的常量的命名规则遵循 PHP 标识符命名规则。const 格式如下:

```
const 常量名=常量值
```

【示例 29】 类的常量的定义。

eg29.php 代码如下。

```
<?php
  class Circle{
    const PI=3.1415926;
    private $radius=3.0;
    public function getArea(){
       return Circle::PI*$this->radius*$this->radius;//可通过"类名::常量名"来访问
                                                     //或通过"self::常量名"来访问
    }
  }
  $c1=new Circle();
  echo $c1->getArea().'<br />';
  echo $c1->PI.'<br />';          //产生一个 Notice,类的常量属于整个类而不属于某个对象
  echo Circle::PI;                //通过"类名::常量名"来访问
  echo PI;                        //产生 Notice,PI 常量在类中定义,因此它属于类
?>
```

说明:在上面的程序中,可以看出类的常量属于某个类而不属于某个具体对象,因此不能通过"对象->常量名"来访问,只能通过"类名::常量名"来访问。在类里面还可以通过"self::常量名"来访问。类的常量的使用需要注意以下几点。

- 在定义时必须设初始值。
- 前面不加任何修饰符。
- 常量名字母一般都用大写。
- 常量可以被子类继承。
- 一个常量属于一个类,而不属于某个对象。

3.4.3　系统常量

为了方便程序设计，在 PHP 中预定义了很多常量，这些常量被称为系统常量。很多这样的常量都定义在扩展库中，因此不能直接使用，加载这样的扩展库后可以使用这些系统常量。常见的系统常量见表 3-3。

表 3-3　系统常量

系 统 常 量	说　　明
__FILE__	当前 PHP 文件的相对路径
__LINE__	当前 PHP 文件中所在的行号
__FUNCTION__	当前的函数名，只对函数内的调用起作用
__CLASS__	当前的类名，只对类起作用
PHP_VERSION	当前使用的 PHP 版本号
PHP_OS	当前 PHP 环境的运行操作系统
TRUE	与 true 一样
FALSE	与 false 一样
M_PI	圆周率常量值
M_E	科学常数 e
M_LOG2E	代表 log2e，即以 2 为底 e 的对数
M_LOG10E	代表 lge，即以 10 为底 e 的对数
M_LN2	2 的自然对数
M_LN10	10 的自然对数
E_ERROR	最近的错误之处
E_WARNING	最近的警告之处
E_PARSE	剖析语法有潜在问题之处
__METHOD__	表示类方法名，比如 B::test

【示例 30】　系统常量举例。

eg30.php 代码如下。

```
<?php
  class Test{
    function show(){
      echo '__CLASS__='.__CLASS__.'<br />';
      echo '__METHOD__='.__METHOD__.'<br />';
    }
  }
  function fun1(){
    echo '__FUNCTION__='.__FUNCTION__.'<br />';
    echo '__METHOD__='.__METHOD__.'<br />';
  }
  $t1=new Test();
  $t1->show();                              //输出为__CLASS__=Test
                                            //输出为__METHOD__=Test::show
```

```
    fun1();                                      //输出为__FUNCTION__=fun1
                                                 //输出为__METHOD__=fun1
    //下一句输出:__FILE__=D:\phpStudy\WWW\ch3\eg30.php
    echo '__FILE__='.__FILE__.'<br />';
    echo '__LINE__='.__LINE__.'<br />';          //输出为__LINE__=19
    echo 'PHP_VERSION='.PHP_VERSION.'<br />';    //输出为 HP_VERSION=5.4.45
    echo 'PHP_OS='.PHP_OS.'<br />';              //输出为 PHP_OS=WINNT
    var_dump(TRUE);                              //输出为 bool(true)
    echo '<br />';
    var_dump(FALSE);                             //输出为 bool(false)
    echo '<br />';
    echo 'M_PI='.M_PI.'<br />';                  //输出为 M_PI=3.1415926535898
    echo 'M_E='.M_E.'<br />';                    //输出为 M_E=2.718281828459
    echo 'M_LOG2E='.M_LOG2E.'<br />';            //输出为 M_LOG2E=1.442695040889
    echo 'M_LOG10E='.M_LOG10E.'<br />';          //输出为 M_LOG10E=0.43429448190325
    echo 'M_LN2='.M_LN2.'<br />';                //输出为 M_LN2=0.69314718055995
    echo 'M_LN10='.M_LN10.'<br />';              //输出为 M_LN10=2.302585092994
    echo 'E_ERROR='.E_ERROR.'<br />';            //输出为 E_ERROR=1
    echo 'E_WARNING='.E_WARNING.'<br />';        //输出为 E_WARNING=2
    echo 'E_PARSE='.E_PARSE.'<br />';            //输出为 E_PARSE=4
?>
```

3.5 运　算　符

在 PHP 程序设计中,任何可以计算出值的“式子”都是表达式,比如常量、变量、函数等都是表达式。而更多的表达式则是由常量、变量和函数再结合一定的运算符构成的。运算符用于决定常量、变量或者函数要进行的操作类型,本节将详细介绍 PHP 中的运算符。

3.5.1 运算符的优先级

运算符的优先级是指在表达式中出现多个运算符时到底先运算哪一个运算符,然后再运算哪一个运算符。在 PHP 程序设计中还有一个概念就是运算符的结合性,所谓运算符的结合性,就是相同优先级的运算符在一起时的计算顺序。结合性分为从左到右和从右到左两种结合性。表 3-4 是 PHP 中运算符的优先级和结合性,这张表摘自《PHP 中文手册》,具备一定的完整性和权威性。

表 3-4　PHP 运算符的优先级和结合性

运　算　符	结合性	说　　明
clone　new	无	clone 和 new
[	左	array()
++　--　~　(int)　(float)　(string)　(array)　(object)　(bool)　@	右	类型和递增/递减
instance of	无	类型
!	右	逻辑运算符

续表

运　算　符	结合性	说　　明
*　/　%	左	算术运算符
+　.	左	算术运算符和字符串运算符
<<　>>	左	位运算符
==　!=　===　!==　<>	无	比较运算符
&	左	位运算符和引用
^	左	位运算符
|	左	位运算符
&&	左	逻辑运算符
||	左	逻辑运算符
? :	左	三元运算符
=　+=　-=　*=　/=　.=　%= &=　|=　^=　<<=　>>=　=>	右	赋值运算符
and	左	逻辑运算符
xor	左	逻辑运算符
or	左	逻辑运算符
,	左	多处用到

3.5.2　算术运算符

在 PHP 中算术运算符是最常见的一种运算符，用于完成各种算术运算。表 3-5 给出了算术运算符的完整描述。

表 3-5　PHP 算术运算符

运算符	名　　称	示　　例	说　　明
+	加	$a+$b	求$a和$b的和
-	减	$a-$b	求$a和$b的差
*	乘	$a*$b	求$a和$b的积
/	除	$a/$b	求$a和$b的商
%	取模	$a%$b	求$a除以$b的余数

【示例 31】　算术运算符举例。

eg31.php 代码如下。

```
<?php
  $a=14.1;
  $b=5.9;
  $c=$a+$b;
  $d=$a-$b;
  $e=$a*$b;
  $f=$a/$b;                    //返回浮点数
  echo $c.'<br />';
  echo $d.'<br />';
```

```
  echo $e.'<br />';
  echo $f.'<br />';
  echo ( 14.1 %  5.9).'<br />';      //先对 14.1 和 5.9 去尾取整再求余数,结果为 4
  echo (-14.1 %  5.9).'<br />';      //先对-14.1 和 5.9 去尾取整再求余数,结果为-4
  echo ( 14.1 %  -5.9).'<br />';     //先对 14.1 和-5.9 去尾取整再求余数,结果为 4
  echo (-14.1 %  -5.9).'<br />';     //先对-14.1 和-5.9 去尾取整再求余数,结果为-4
?>
```

说明：除法运算“/”得到的总是浮点数;取模运算总是把操作数先去尾取整再取模,得到的余数的符号要么是 0,要么与第一个操作数符号相同。

3.5.3 赋值运算符

赋值运算就是把表达式的值赋给变量的过程。PHP 中的最基本的赋值运算符是“＝”,此外像＋＝、－＝、＊＝、\＝、%＝、.＝等也是赋值运算符。

特别需要注意的是,像“ $a 赋值运算符 ＝ 表达式”表示的含义是“ $a ＝ $a 赋值运算符(表达式)”,也就是先运算“表达式”,再进行赋值运算,这点非常容易错误理解。表 3-6 是 PHP 赋值运算符。

表 3-6 PHP 赋值运算符

运算符	名 称	示 例	说 明
＝	等于	$a=1	$a ＝ 1
＋＝	加等于	$a ＋＝ 3＋4	$a ＝ $a ＋ (3＋4)
－＝	减等于	$a －＝ 3－4	$a ＝ $a － (3－4)
＊＝	乘等于	$a ＊＝ 3＋4	$a ＝$a ＊ (3＋4)
\＝	除等于	$a /＝ 6;	$a ＝ $a /6
%＝	取模等于	$a %＝ 4;	$a ＝$a %4
.＝	连接等于	$a .＝ 'abc'	$a ＝$a.'abc'

【示例 32】 赋值运算符举例。

eg32.php 代码如下。

```
<?php
  $a=20;
  $a +=30;
  echo $a.'<br />';                  //输出 50
  $a -=5;
  echo $a.'<br />';                  //输出 45
  $a *=3+2;
  echo $a.'<br />';                  //输出 225。先计算右边表达式,即 45*(3+2)
  $a /=5;
  echo $a.'<br />';                  //输出 45
  $a %=2;
  echo $a.'<br />';                  //输出 1
  $b='I love ';
  $b .='China.';
  echo $b.'<br />';                  //输出“I love China”
```

```
  $c=2;
  $c *=$c +=3;                    //先计算$c +=3,得到$c=5;再计算 $c *=5,得到 25
  echo $c.'<br />';               //输出 25
?>
```

说明：

- 上述代码在执行“$a *= 3+2”时，先执行 3+2 得到 5，再执行 $a *= 5，即 $a=45 * 5=225；
- 在执行语句“$c *= $c += 3”时，先执行“$c += 3”得到 $c=5，再执行“$a *= 5”得到 $a=5 * 5，即结果为 25。

3.5.4　比较运算符

比较运算符用于比较两个或多个表达式的大小，最后根据比较结果返回一个布尔值。一般在程序中常常用于分支控制结构，通过比较结果来决定执行哪一段代码。

数值表达式在比较大小时，由数值的大小决定；字符串的大小由字符串在英文字典中出现的先后顺序来决定，在字典中后出现的字符串大于先出现的字符串；布尔类型中逻辑真的值(true)大于逻辑假(false)。表 3-7 是 PHP 比较运算符。

表 3-7　PHP 比较运算符

运算符	名　称	示　　例	说　　明
==	等于	0==0,0==false	两个都成立，结果为 true
===	全等于	0===0, 0===false	全等运算符只有在运算符两边表达式的值和类型都相同时才返回 true，因此 0=== false 的值是 false，而 0===0的值是 true
!=	不等于	$a != $b	二者不相等时，返回 true
<>	不等于	$a<>$b	二者不相等时，返回 true
!==	非全等		
<	小于	$a<$b	$a 小于 $b 时，返回 true
<=	小于等于	$a<=$b	$a 小于或者等于 $b 时，返回 true
>	大于	$a>$b	$a 大于 $b 时，返回 true
>=	大于等于	$a>=$b	$a 大于或等于 $b 时，返回 true

【示例 33】　比较运算符举例。

eg33. php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $a=5;
  $b=4;
  var_dump($a>$b);                //true
  $st1='liubei';
  $st2='Liubei';
  var_dump($st1>$st2);            //true
  $st3='liubei';
  $st4='liu';
```

```
    var_dump($st3>$st4);              //true
    $b1=true;
    $b2=false;
    var_dump($b1>$b2);                //true
    $str='bde';
    $in='bd';
    echo '<br />';
    if (strpos($str,$in) ===false)    //不能写成 ==
      echo "{$in}不是{$str}的子串";
    else
      echo "{$in}是{$str}的子串";
?>
```

说明：在判断子串时不能写成“==”,一定要写成“===”,因为当 $in 在 $str 中出现的位置是 0 时,0==false 的值是 true,而 0===false 的值是 false。strpos()函数详解见后面章节。

3.5.5 三元运算符

三元运算符有 3 个操作数,是 PHP 中唯一的有 3 个操作数的运算符,它的格式如下。

```
<表达式 1>?<表达式 2>:<表达式 3>
```

说明：若表达式 1 非 0(包括非 NULL、非 false、非空),则返回表达式 2,否则返回表达式 3;在实际应用中表达式 1 一般是布尔型的表达式。

【示例 34】 三元运算符举例。

eg34.php 代码如下。

```
<?php
  $a =3;
  $b =4;
  $c =5;
  $d =$a ?$b : $c;                        //$a 非 0时返回$b
  var_dump($d);                           //int(4)
  $num1 =5;
  $num2 =6;
  $max = ($num1>$num2) ?$num1 : $num2;    //求最大值
  echo $max;
?>
```

3.5.6 逻辑运算符

逻辑运算符用于将两个表达式比较的结果转换为布尔值,因此又称为布尔运算符。逻辑运算符常常用于分支结构的程序设计。表 3-8 给出了 PHP 逻辑运算符。

表 3-8 PHP 逻辑运算符

运算符	名 称	示 例	说 明
AND	逻辑与	$a AND $b	如果 $a 和 $b 都为 true,则返回 true,否则返回 false

续表

运算符	名　称	示　　例	说　　明
&&	逻辑与	$a && $b	如果 $a 和 $b 都为 true，则返回 true，否则返回 false
OR	逻辑或	$a OR $b	如果 $a 和 $b 中全为 false，则返回 false，否则返回 true
||	逻辑或	$a || $b	如果 $a 和 $b 中全为 false，则返回 false，否则返回 true
!	逻辑非	!$a	$a 为 true 时返回 false，$a 为 false 时返回 true
XOR	逻辑异或	$a XOR $b	$a 和 $b 同为 true 或同为 false 时返回 false，否则返回 true

【示例 35】 逻辑运算符举例。

eg35.php 代码如下。

```
<?php
  $b1 = $b2 = true;
  $b3_1 = $b1 && $b1;
  $b3_2 = $b1 AND $b1;
  var_dump($b3_1,$b3_2);     //true,true
  $b4 = false;
  $b5 = true;
  $b6_1 = $b1 || $b1;
  $b6_2 = $b1 OR $b1;
  var_dump($b6_1,$b6_2);     //true,true
  $b7 = false;
  $b8 = true;
  $b9_1 = $b7 XOR $b8;
  $b9_2 = $b7 XOR $b7;
  var_dump($b9_1,$b9_2);     //false,false
  $b10 = true;
  $b11 = false;
  $b12_1 = !$b10;
  $b12_2 = !$b11;
  var_dump($b12_1,$b12_2);   //false,true
?>
```

3.5.7　运算符的“短路”

PHP 在进行逻辑运算时，如果表达式的值已经可以确定，则运算不会往下继续进行，这就是逻辑运算符的短路。or、||、and、&& 都是短路运算符。短路的原则如下。

- “条件 a” && “条件 b”：当条件 a 为假时，条件 b 不再执行。
- “条件 a” || “条件 b”：当条件 a 为真时，条件 b 不再执行。
- “条件 a” and “条件 b”：当条件 a 为假时，条件 b 不再执行。
- “条件 a” or “条件 b”：当条件 a 为真时，条件 b 不再执行。

说明：以“条件 a” && “条件 b”为例。如果条件 a 为 false，则整个表达式的值一定为 false，此时没有必须运算条件 b，因此发生短路。其他几个短路原则道理相同。

【示例 36】 逻辑运算符的短路。

eg36.php 代码如下。

```
<?php
  $a=1;
  $b=2;
  $c=3;
  $d=($a<$b) || ($c=30);       //因$a<$b成立,则$d为 true,无须继续给$c赋值
  var_dump($d);                //true
  var_dump($c);                //3 而不是 30
  $e=($a>$b) && ($a=2);        //因$a>$b不成立,则$e为 false,无须继续给$a赋值
  var_dump($e);                //false
  var_dump($a);                //1
?>
```

在 PHP 中短路运算常常可以让程序更灵活,例如:“@fopen("readme.txt") or die("文件不存在");”,该语句打开文件 readme.txt,文件若存在则不执行后面的 die 命令,文件若不存在则执行 die 命令,让程序“死掉”,不再向下执行。再例如“$con = mysql_connect("127.0.0.1","root","root") or die('Could not connect:'. mysql_error());”,该语句用于链接数据库服务器,若链接成功则程序不会“死掉”;否则程序会“死掉”,防止程序继续向下执行一些毫无意义的操作。

3.5.8 位运算符

位运算符将一个整数表达式看作一系列的二进制位(bit)来处理。常常在加密或者用户权限处理时用到。PHP 位运算符如表 3-9 所示。

表 3-9 PHP 位运算符

运算符	名 称	示 例	说 明
&	按位与	$a & $b	将把 $a 和 $b 中都为 1 的位设为 1
|	按位同或	$a | $b	将把 $a 和 $b 中任何一个为 1 的位设为 1
^	按位异或	$a ^ $b	将把 $a 和 $b 中一个为 1、另一个为 0 的位设为 1
~	按位取反	~ $a	将 $a 中为 0 的位设为 1,反之亦然
<<	左移	$a << $b	将 $a 中的位向左移动 $b 次(每一次移动都表示“乘以 2”)
>>	右移	$a >> $b	将 $a 中的位向右移动 $b 次(每一次移动都表示“除以 2”)

【示例 37】 位运算符举例。

eg37.php 代码如下。

```
<?php
  echo '7 & 15 ='.(7 & 15).'<br />';      //输出:7 & 15 =7
  echo '7 | 15 ='.(7 | 15).'<br />';      //输出:7 | 15 =15
  echo '7 ^ 15 ='.(7 ^ 15).'<br />';      //输出:7 ^ 15 =8
  echo '~ 7 ='.( ~ 7 ).'<br />';          //输出:~7 =-8
  echo '8 <<2 ='.(8 <<2).'<br />' ;       //输出:8 <<2 =32
  echo '7 >>2 ='.(7 >>2).'<br />';        //输出:7 >>2 =1
?>
```

说明:下列算式中等号左边为结果,右边为第 1 个和第 2 个操作数。

```
( 7=00000111)= (00000111) & (00001111)
```

```
(15=00001111)=(00000111) | (00001111)
( 8=00001000)=(00000111) ^ (00001111)
(-8=11111000)=~(00000111)
(32=00100000)=(00001000) << (00000010)   //左移 2 位,右边补 0,相当于变大 4 倍
(1=00000001)=(00000111) >> (00000010)    //右移 2 位,左边补 0,相当于去掉右边两个 1
```

3.5.9　递增和递减运算符

在 PHP 中递增和递减运算符用于变量的自动加 1 或者自动减 1,这种运算符可以使代码更加简洁。递增运算符有两种方式,即++ $a 和 $a++,对应于递减的就是-- $a 和 $a--。但是++和--放置在变量的前面和放置在变量的后面又有区别。以++为例:

```
<?php
  $a=1;
  $b=++$a;
  echo '$a='.$a.', $b='.$b.'<br />';     //输出:$a=2, $b=2
  $c=1;
  $d=$c++;
  echo '$c='.$c.', $d='.$d.'<br />';     //输出:$c=2, $d=1
?>
```

说明:在“$b = ++$a”中,先将 $a 加 1,再把 $a 的值赋给 $b,这样 $a 和 $b 是相等的;而在“$d = $c++”中,先把 $c 的值赋给 $d,再把 $c 加 1,因此 $d 是比 $c 小的。它们的运算规则如表 3-10 所示。

表 3-10　PHP 递增和递减运算符运算规则

运　算　符	名称	返　回　值	说　　明
$ variable++	递增	$ variable	先参与其他运算,再把 $ variable 加 1
++ $ variable	递增	$ variable+1	先把 $ variable 加 1,再参与其他运算
$ variable--	递减	$ variable	先参与其他运算,再把 $ variable 减 1
-- $ variable	递减	$ variable-1	先把 $ variable 减 1,再参与其他运算

【示例 38】　递增和递减运算符举例。

eg38.php 代码如下。

```
<?php
  $a=1;
  $b=++$a +++$a +++$a +$a;            //理解为: 2+3+4 +4 =13
  echo '$a='.$a.',$b='.$b;            //输出: $a=4, $b=13
?>
```

说明:在书写代码时,像类似“$b=++ $a+++ $b;”的写法容易出错,应该把“+++”分开写成形如“$b=++ $a+ ++ $b;”的形式,这样系统才能正常解释。实际上系统会把“$b= $a+++ $b;”理解为“$b= $a++ + $b;”。现实程序设计中应有意去避免这种情况,主动用空格隔开,或者人为加括号改变递增和递减的结合位置,避免一些不必要的麻烦。

3.5.10 执行运算符

PHP 支持一个所谓的执行运算符“`”(在 Esc 键下面),这个不是单引号。PHP 尝试将执行运算符中的内容作为操作系统命令来执行,并输出执行结果。它的执行效果和 shell_exec()函数相同。但是执行运算符在激活了安全模式或者关闭了 shell_exec()函数时是无效的。

【示例 39】 执行运算符举例。

eg39.php 代码如下。

```
<?php
  $cmd = `cmd`;
  $dir = `dir`;
  echo $cmd;
  echo "<pre>$dir</pre>";
?>
```

程序运行结果如图 3-10 所示。

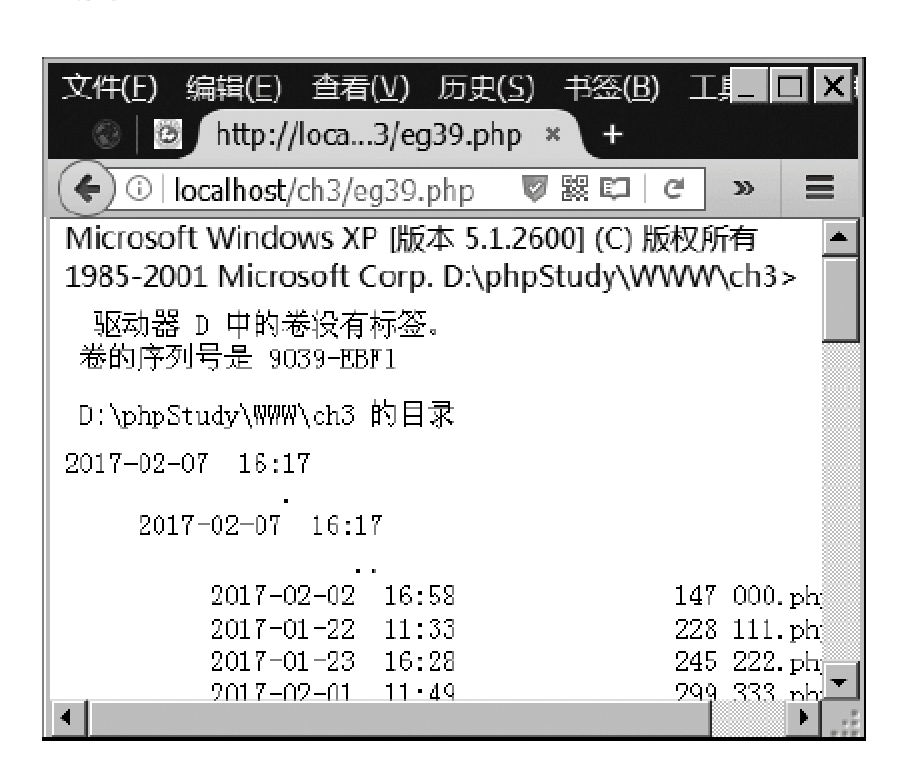


图 3-10 执行运算符

3.5.11 错误控制运算符

PHP 支持一个所谓的错误控制运算符@,若将@符号放置在 PHP 表达式之前,则该语句可能产生的任何错误(这里说的错误是广义的错误,包括 Notice、Warning 和 Error)信息都会被忽略。如果激活了 track_errors 属性,则语句产生的错误信息被存放在 $php_errormsg 变量中。此变量在每次出错时都会被覆盖,因此如果想了解错误信息,应及时检查 $php_errormsg 的值。

错误控制运算符一般用于可以忽略的脚本中的可有可无的错误信息,例如希望在输出一个变量时,如果变量存在则输出,不存在时也不要显示“变量不存在”的信息。

```
<?php
  echo $a;
  echo @$a;
?>
```

说明：前者代码会显示错误信息“Notice：Undefined variable：a in D:\phpStudy\WWW\ch3\eg40.php on line 2”，而后者不显示错误信息。

【示例 40】 在数据库程序设计的分页显示时经常要从网址获取当前是第几页的信息。在网址不包含页面信息时容易出错，此时使用错误控制运算符比较好。

eg40.php 代码如下。

```
<?php
  @$pageno =$_GET['pageno'] ?$_GET['pageno'] : 1;
  echo 'pageno ='.$pageno;
?>
```

程序运行结果如图 3-11 和图 3-12 所示。



图 3-11　网址不含页面信息

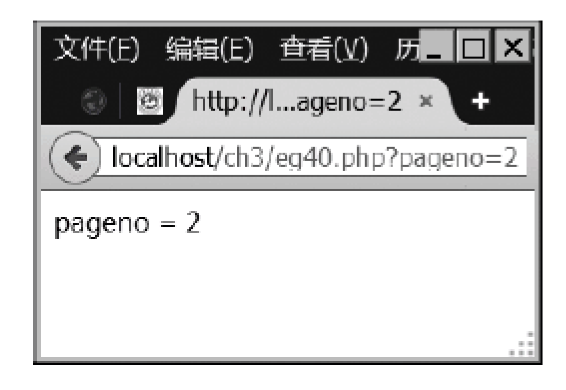


图 3-12　网址含页面信息

说明：

- 在输入网址 http://localhost/ch3/eg40.php 时，由于网址不含页面信息，会产生 Notice 类型的“错误信息”。此时语句“@ $pageno = $_GET['pageno'] ? $_GET['pageno'] : 1;”中的 $_GET['pageno']的值为 NULL，表达式“ $_GET['pageno']? $_GET['pageno'] : 1”的值为 1。
- 在输入网址 http://localhost/ch3/eg40.php? pageno=2 时，由于网址中包含页面信息，不会产生任何错误信息。此时语句“@ $pageno = $_GET['pageno'] ? $_GET['pageno'] : 1;”中的 $_GET['pageno']的值为 2，因为 $_GET['pageno']存在，表达式“ $_GET['pageno']? $_GET['pageno'] : 1”中的“ $_GET['pageno']”的值为 2。

3.5.12　PHP 表达式

表达式是 PHP 最重要的基石。在 PHP 中，几乎所写的任何东西都是一个表达式。简单但却最精确的定义一个表达式的方式就是“任何有值的东西”。最基本的表达式形式是常量和变量。当输入“ $ var = 15”，即将值 15 分配给变量 $ var。很明显，"15"的值为 15，换句话说，"15"是一个值为 15 的表达式(在这里，"15"是一个整型常量)。赋值之后，所期待情况是 $ var 的值为 15，因而如果写下 $ a = $ var，期望的是它犹如 $ a = 15 一样。也就是说，$ var 是一个值也为 15 的表达式。如果一切运行正确，那这正是将要发生的正确结果。稍微复杂的表达式例子就是函数。例如，考虑下面的函数：

```
<?php
  function  foo(){
   echo 'Hello, PHP!';
```

```
    return  15 ;
  }
?>
```

3.6 综合案例——短路运算和优先级

在 PHP 等程序设计语言中，逻辑表达式或者关系表达式在计算出结果之后不会继续向后运算，后面的运算不会被执行，称为短路运算。

例如“ $ b = false && foo();”，无论 foo()函数返回值为多少，都不会改变 $ b 值为 false 的事实，因此程序不会去调用或执行 foo()函数。

而在语句“ $ x = false or true;”中，赋值运算优先级高于逻辑运算，因此原语句等价于“($ x = false)or true;”，故 $ x 值为 false。

步骤 1 新建 eg41. PHP 页面，输入以下代码：

```
<?php
  echo "<pre>";
  //--------------------
  //foo() 根本没有机会被调用，被运算符“短路”了
  $a  =( false  &&  foo ());  //根据 && 的运算规律，$a 必等于 false，故 foo()函数不执行
  $b  =( true   ||  foo ());  //根据||的运算规律，$b 必等于 true，故 foo()函数不执行
  $c  =( false  and  foo ());//根据 and 的运算规律，$c 必等于 false，故 foo()函数不执行
  $d  =( true   or   foo ());  //根据 or 的运算规律，$d 必等于 true，故 foo()函数不执行
  //--------------------
  //“||”比“or”的优先级高
  //表达式 (false || true) 的结果被赋给 $e
  //等同于：($e = (false || true))
  $e  =  false  ||  true ;
  //常量 false 被赋给 $f，true 被忽略
  //等同于：(($f =false) or true)
  $f  =  false  or  true ;
  var_dump ( $e,  $f );
  //--------------------
  //“&&” 比 “and” 的优先级高
  //表达式 (true && false) 的结果被赋给 $g
  //等同于：($g = (true && false))
  $g  =  true  &&  false ;
  //常量 true 被赋给 $h，false 被忽略
  //等同于：(($h =true) and false)
  $h  =  true  and  false ;
  var_dump ( $g,  $h );
?>
```

步骤 2 运行该程序，程序运行结果如图 3-13 所示。



图 3-13　短路运算

3.7　习　　题

一、填空题

1. 四种标量数据类型是________、________、________、________。
2. 声明全局变量使用的关键字是________。
3. 强制数据类型转换使 $a 为整型变量，$a=________7.9。
4. 自动类型转换。$f = '1'；$f = 1 ________ $f。使 $f 的值为 2。
5. 数据类型函数。"$b = 'true'；echo gettype($b)；"的输出为________。
6. "$a = 4；$a =________；$b =5；echo $a；"执行前面的一些语句后输出 5。
7. "$a = 'Changsha'；$ab = 'Wuhan'；echo "$ab"；"的输出为________。echo "{$a}b" 的输出为________。echo '$ab'；的输出为________。
8. 销毁变量可以是________函数。
9. 在类中定义常量使用关键字________。
10. 执行"$a = 1；$b = 1；$c = $a == $b；"后 $c 的值为________。
11. 执行"$a= 4；$a *= $a + 6；echo $a；"后输出结果为________。
12. 执行"$a = 1；$b = true；$c = $a === $b；"后 $c 的值为________。
13. 执行"$a = 1；$b = 5；$c = $a > $b? $a : $b；"后 $c 的值为________。
14. 执行下面的代码

```
<?php
  $a=1;
  $b=2;
  if (($a<$b) || ($a=5))
    echo $a."<br />";
  $a=1;
  $b=2;
  if (($a<$b) && ($a=5))
    echo $a."<br />";
?>
```

输出结果为________和________。

二、选择题

1. 下面选项中合法的变量是________。

   A. $ a_1　　B. $ a^3　　C. $ stu-name　　D. $ 66a

2. 关于 define()函数与 const 关键字的不同点，下面说法正确的是________。

   A. define()函数和 const 关键字定义的常量都不区分大小写

   B. define()函数可以用于类成员变量的定义，不可修改；const 关键字可以用于定义全局变量

   C. define()函数可以采用表达式作为常量名称；const 采用一个普通的常量名称

   D. define()函数和 const 都可以在条件语句中定义常量

3. 下列几个运算符，优先级别最高的是________。

   A. 算术运算符　　B. 关系运算符　　C. 赋值运算符　　D. 逻辑运算符

4. 下面运算符中，不属于比较运算符的是________。

   A. ==　　B. >=　　C. =　　D. !==

5. 下面的程序运行结果是________。

```
<?php
  $a=1;
  $b=2;
  if (($a>$b) || ($a=5) || ($b=6))
    echo $a."  ".$b;
?>
```

   A. 没有输出　　B. 1　2　　C. 5　2　　D. 5　6

6. 下面的程序运行结果为________。

```
<?php
  $a=1;
  $b=2;
  $c=++$a+++$b;
  echo $c;
?>
```

   A. 3　　B. 4　　C. 5　　D. 6

7. 下面的程序运行结果为________。

```
<?php
  $a=1;
  echo $a."  ";
  echo @$b;
?>
```

   A. 1　0

   B. 1　false

   C. 1

   D. 1　Notice：Undefined variable：b in … on line 4

三、程序设计题

1. 编写程序，设计多个常量、变量，并为这些常量、变量赋值。

2. 编写程序，设计多个常量、变量，例如圆的半径、圆周率常量，计算圆的面积和周长。

3. 编写程序，要求用到“与”和“或”运算的短路运算。

第 4 章　PHP 流程控制语句

知识点：

- 语句的概念
- if 语句和 if...else 语句
- if 语句的嵌套
- switch...case 语句
- for 循环语句
- while 和 do...while 循环语句
- continue 和 break 语句
- foreach 语句

本章导读：

语句是程序完成一定的操作的最基本的单元。像其他语言一样，PHP 也分为 3 种程序结构，即顺序结构、分支机构和循环结构。每一个 PHP 程序都是由若干语句构成的，顺序结构是指程序从上到下逐句执行，不存在分支和循环，顺序结构是最基本的和常见的程序结构，这里无须赘述。

下面我们重点来讲述分支结构和循环结构。

4.1　分 支 结 构

所谓分支结构，是指程序向下执行的过程中需要根据某个条件或者运行结果来选择不同的执行语句。例如登录程序，当用户名和密码都正确时可以进入系统，而用户名或者密码错误则不允许进入系统。

在 PHP 中提供了以下几种分支结构。

- if 语句；
- if...else 语句；
- if...elseif ...else 语句；
- switch...case 语句。

4.1.1　if 语句

if 语句是最简单的分支结构语句，用于在条件满足时执行 if 后面的语句。if 语句语法如下。

```
if (条件)
```

```
{ 语句块 }
```

if 语句在执行时，首先判断条件是否为 true，若为 true 则执行“语句块”，然后结束整个 if 语句；若为 false，则不执行语句块，直接结束整个 if 语句。if 语句执行流程如图 4-1 所示。

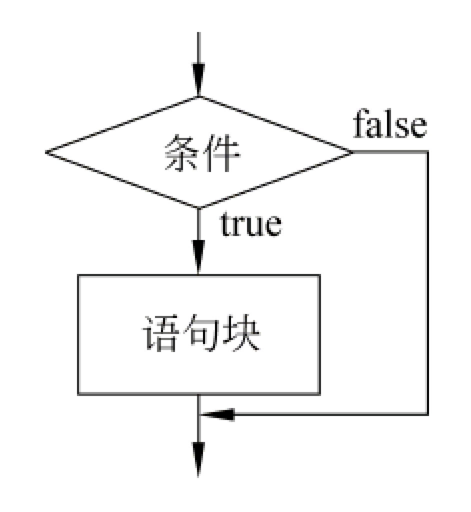


图 4-1　if 语句执行流程

说明：若语句块是 1 个语句时，大括号可以省略不写；若语句块是多个语句，则语句之间用分号隔开，此时大括号不能少。例如：

```
if ($a>$b){
  echo $a;
}
```

可以写成

```
if ($a>$b)
  echo $a;
```

而

```
if ($a>$b){
  echo $a;
  echo $b;
}
```

则不能写成

```
if ($a>$b)
  echo $a;
  echo $b;
```

此时若不写大括号，则语句“echo $b;”不受 if 条件控制。此外，if 条件小括号之后千万不要加“;”，若加了“;”，则后面的语句块不受 if 语句的控制，系统会把“分号”理解为空语句，因此 if 语句控制的是空语句。即：

```
if ($a>$b)
  echo $a;
```

不能写成

```
if  ($a>$b);
  echo $a;        //本语句将不受 if 条件控制，会无条件执行
```

【示例 1】 if 语句举例。

eg1.php 代码如下。

```
<?php
  $city1 = 'Beijing';
  if ($city1 == 'Beijing')   //"=="用于判断是否相等
    echo "$city1 is the capital of China.<br />";
  $city2 = 'Wuhan';
  if ($city2 = 'Changsha')   //"="用于赋值，$city2 被赋值为 'Changsha'，同时 if 条件成立
```

```
    echo "$city2 is the capital of Hunan.<br />";
  $city3 = 'Hefei';
  if ($city3 == 'Nanjing'); //条件后加分号,后面的语句不受此 if 控制,将无条件执行
  echo "$city3 is the capital of Jiangsu.<br />";
?>
```

程序运行结果如图 4-2 所示。



图 4-2　if 语句运行举例

【示例 2】 求两个数的最大值。

eg2.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $a = 1;
  $b = 2;
  if ($a > $b)
    $max = $a;
  if ($a <= $b)
    $max = $b;
  echo "最大数是:$max";
?>
```

4.1.2　if...else 语句

if...else 语句在 if 语句的基础上添加了当条件不满足时要进行的操作。if...else 语句的语法格式如下。

```
if (条件){
  语句块 1
}else{
  语句块 2
}
```

if...else 语句首先要判断 if 条件,当条件为 true 时,执行语句块 1,然后结束整个 if...else 语句;当条件为 false 时,执行语句块 2,然后结束整个 if...else 语句。if...else 语句执行流程如图 4-3 所示。

说明:语句块 1 和语句块 2 可能是 1 个或多个语句。如果是 1 个语句,则大括号可以省略,否则不能省略大括号。

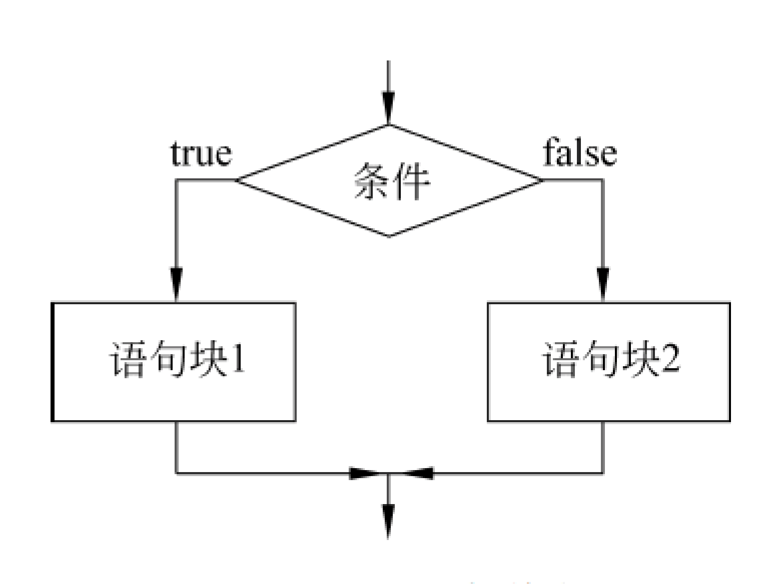


图 4-3　if...else 语句执行流程

【示例 3】 求两个数的最大值。

eg3.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $a=1;
  $b=2;
  if ($a>$b)
    $max=$a;
  else
    $max=$b;
  echo "最大数是:$max";
?>
```

4.1.3　if...elseif...else 语句

if...elseif...else(也可以写成 if...else if...else)可以提供更多分支,将数据依次分类排除。if...elseif...else 语法格式为:

```
if(条件 1)
{ 语句块 1 }
[elseif  (条件 2)
{ 语句块 2 }
[elseif  (条件 3)
{ 语句块 3 }
...
[elseif  (条件 n)
{ 语句块 n }
[else
{ 语句块 n+1 }]]...]]
```

程序首先判断条件 1,若条件 1 为 true 则执行语句块 1,然后结束整个 if...elseif...else 语句;若条件 1 为 false 则判断条件 2。若条件 2 为 true 则执行语句块 2,然后结束整个 if...elseif...else;若条件 2 为 false 则判断条件 3。……若条件 n 为 true 则执行语句块 n,否则执行语句块 $n+1$。if...elseif...else 语句执行流程如图 4-4 所示。

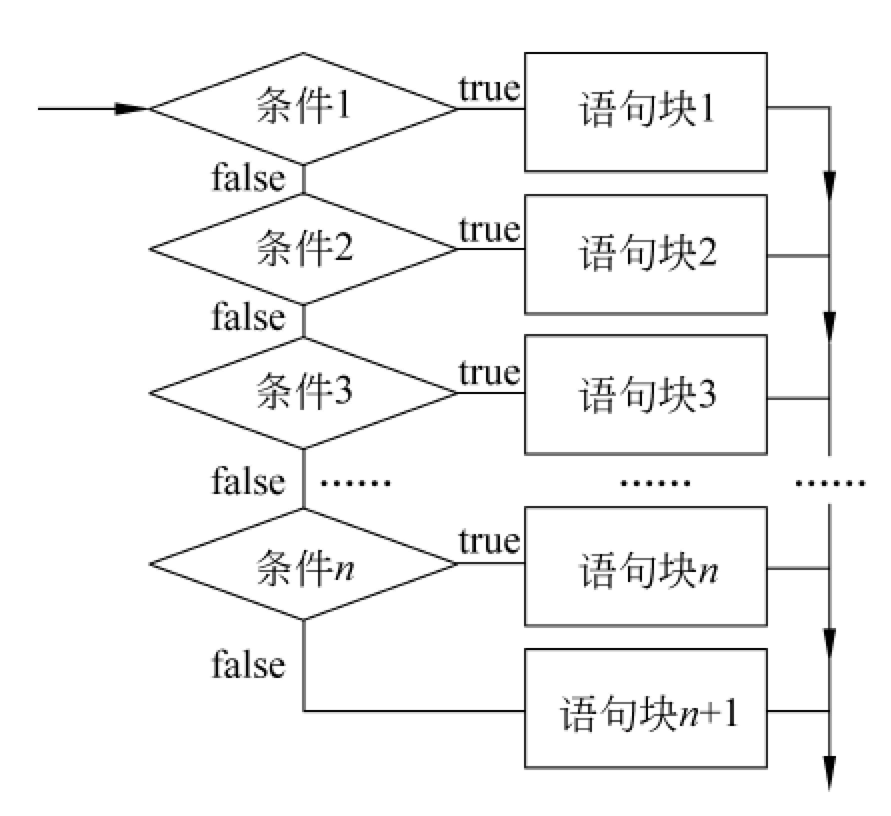


图 4-4　if...elseif...else 语句执行流程

说明：除 if(条件 1)和语句块 1 必需外，其余部分均为可选项；语句块是 1 个语句时大括号可以省略；语句块 1 执行的条件是条件 1 成立，语句块 $n(n>1)$执行的条件是条件 1、条件 2、……、条件 $n-1$ 都不成立而条件 n 成立，语句块 $n+1$ 执行的条件是全部条件都不成立。

【示例 4】 根据分数给出学生成绩的等级。

eg4.php 代码如下。

```
<?php
  $score = 89;
  $grade = '';
  if ($score ==100){
     $grade = 'outstanding';
  }elseif ($score >=85) {
     $grade = 'excellent';
  }elseif ($score >=70){
     $grade = 'good';
  }elseif ($score >=60){
     $grade = 'pass';
  }else{
     $grade = 'fail';
  }
  echo $grade;
?>
```

4.1.4 if 语句的嵌套

if 语句或 if...else 语句的各个语句块又有可能是 if 语句或 if...else 语句，这就是 if 语句的嵌套。在 PHP 中 if 语句的嵌套遵循就近原则：else 总是属于前面离自己最近的那个 if。例如：

```
if (条件 1)
  if (条件 2)
    语句;
  else   //因为离第 2 个 if 近，所以属于第 2 个 if
    语句;
else     //虽然离第 2 个 if 近，但是第 2 个 if 有自己的 else，所以本 else 属于第 1 个 if
  语句;
```

在实际编程中可以通过加大括号来改变 else 的归属，例如：

```
if (条件 1)
  if (条件 2)
    echo $a;
  else   //属于第 2 个 if 语句
    echo $b;
```

加大括号后 else 的归属将发生改变，代码如下。

```
if (条件 1){
  if (条件 2)
```

```
    echo $a;}
else     //属于第 1 个 if 语句
  echo $b;
```

4.1.5　switch…case 语句

switch…case 语句用于多分支结构，它把一个表达式的值和多个表达式进行比较，并根据比较结果来执行不同的语句块。switch…case 语句的格式如下。

```
switch (表达式){
   case 表达式 1:语句块 1;[break;]
   [case 表达式 2:语句块 2;[break;]
   [case 表达式 3:语句块 3;[break;]
     …
   [case 表达式 n:语句块 n;[break;]
   [defalut:语句块 n+1;[break;]]]…]]
}
```

switch…case 语句流程如图 4-5 所示。

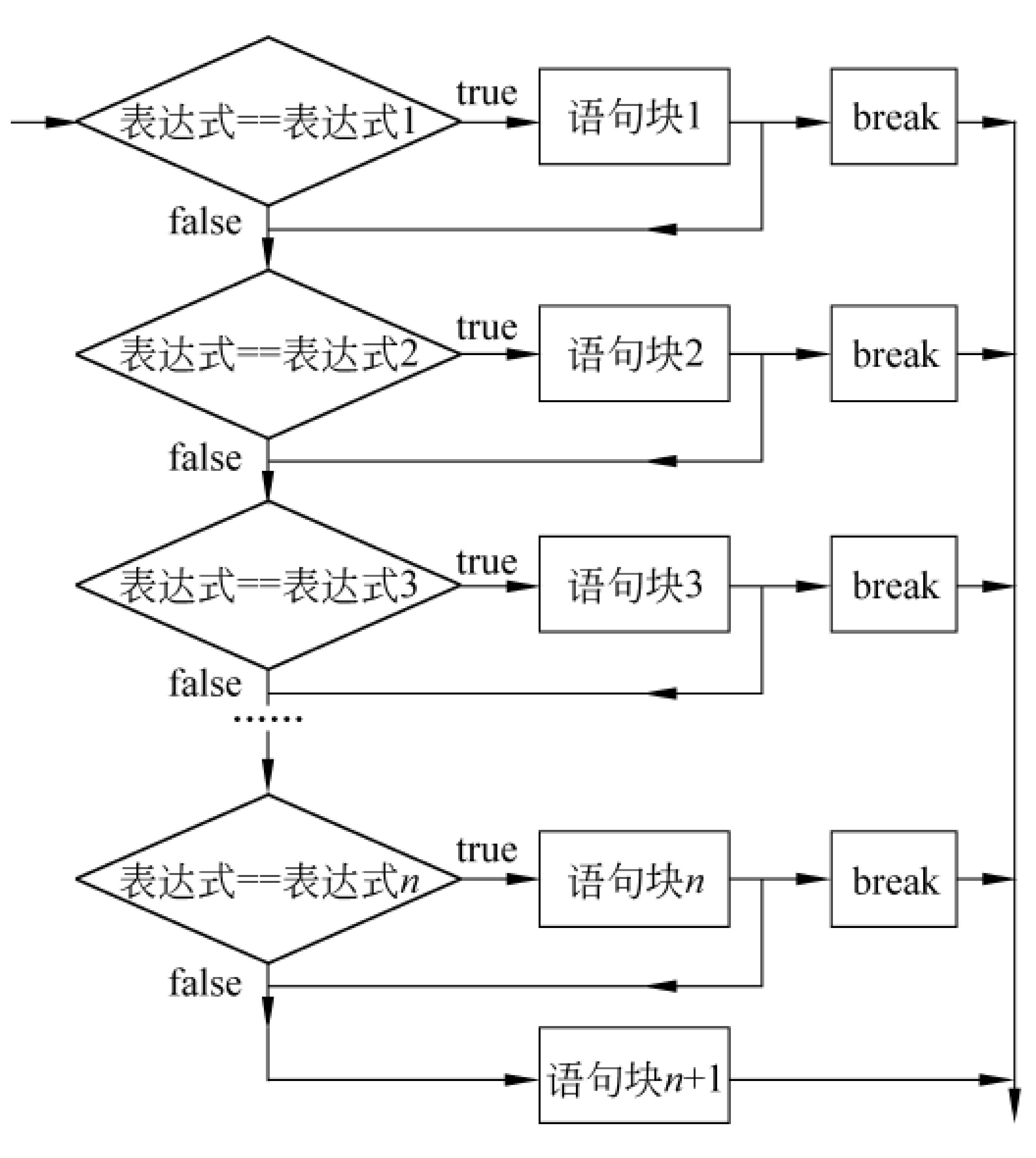


图 4-5　switch…case 语句执行流程

从图 4-5 可以看出：程序首先计算表达式，若表达式的值等于表达式 1 的值，则执行语句块 1；如果有 break 语句，则结束整个 switch…case 语法结构；如果没有 break，则继续执行语句块 2、语句块 3、……、语句块 n 和语句块 $n+1$。若表达式的值不等于表达式 1 的值，则判断表达式的值是否等于表达式 2 的值，若等于，则执行语句块 2；如果有 break，则结束整个 switch…case 语法结构；如果没有 break，则继续执行语句块 3、语句块 4、……、语句块 n 和语句块 $n+1$；以此类推。如果表达式的值不等于任何一个 case 表达式的值，则执行 defalut 后面的语句块 $n+1$。要强调的是，在任何时候执行到 break，都会结束整个 switch…case 语法结构。

说明：

- 表达式 1、表达式 2、……、表达式 *n* 一般为常量；
- 语句块 1、语句块 2、……、语句块 *n*、语句块 *n*+1 可以是 0 个、1 个或多个语句，因为程序是按顺序执行的无论语句块是几个语句都无须加大括号；
- 如果表达式的值等于表达式 *k* 的值，则程序执行语句块 *k*，以及后面的全部语句块，不会再次判断表达式的值是非等于表达式 *k*+1、表达式 *k*+2、……、表达式 *n* 的值；
- 任何时候执行到 break 语句，结束整个 switch...case 语法结构。

【示例 5】 根据月份判断季节。

eg5. php 代码如下。

```
<?php
  $month = 9;
  switch ( $month ){
    case 3:
    case 4:
    case 5: $season = 'Spring'; break;
    case 6:
    case 7: $season = 'Summer'; break;
    case 8: $season = 'Summer'; break;
    case 9:
    case 10:
    case 11: $season = 'Autumn'; break;
    case 12:
    case 1:
    case 2: $season = 'Winter'; break;
    default:$season = 'Error';
  }
  echo "Now the season is $season.";
?>
```

说明：当 $month 值为 3 或者 4 时，程序执行后面的空语句，然后不会再次判断 $month 的值是非为 5，而是直接执行“$season = 'Spring'; break;”，执行到 break 时退出整个 switch...case 语句。当 $month 值不是一个存在的月份时，程序执行 default 后面的语句“$season = 'Error';”。

【示例 6】 判断某年月日是这一年的第几天。

分析：“$year 年 $month 月 $day 日”是这一年的第几天可以理解为两部分之和：第 1 部分为 $day，第 2 部分为前面 $month－1 个月的天数之和。2 月的天数与是否是闰年有关。闰年的条件有两个：年数能被 400 整除；或年数能被 4 整除，但不能被 100 整除；满足两个条件中的任何一个条件即为闰年。

根据分析，eg6. php 代码如下。

```
<?php
  $year = 1900;
  $month = 12;
  $day = 31;
  $days = $day;                    //$days 的第 1 部分
```

```
  $flag = 0;                              //假设是平年
  if ( $year % 400 == 0)                  //满足该条件就是闰年
    $flag = 1;
  if (( $year % 4 == 0) && ( $year % 100 <> 0 ))    //满足该条件就是闰年
    $flag = 1;
  switch ( $month-1 ){                    //$days 的第 2 部分
//没有 break,$month-1 等于某值之后不会继续判断下一表达式,而是一直向下执行
    case 11: $days = $days + 30;
    case 10: $days = $days + 31;
    case 9: $days = $days + 30;
    case 8: $days = $days + 31;
    case 7: $days = $days + 31;
    case 6: $days = $days + 30;
    case 5: $days = $days + 31;
    case 4: $days = $days + 30;
    case 3: $days = $days + 31;
    case 2: $days = $days + 28 + $flag;
    case 1: $days = $days + 31;
  }
  echo "The day is the $days of the year.";
?>
```

程序运行结果如图 4-6 所示。



图 4-6　判断某年月日是这一年的第几天

说明：

- 闰年时令 $flag=1,平年时令 $flag=0;因此 2 月的天数就是 28+ $flag。
- “year 年 month 月 day 日”是这一年的第几天可以理解为两部分之和：第 1 部分为 day,第 2 部分为前面 month−1 个月的天数之和。例如,2010 年 6 月 13 日是这一年的第几天可以理解为 13(13 日)再加前 6 个月减去 1 个月的天数之和,即 1 月、2 月、3 月、4 月和 5 月每个月的天数之和。语句块中没有 break 语句,程序会从 month−1 月开始相加,一直加到 1 月刚好把前 month−1 个月的天数全部加在一起。

4.2　循 环 结 构

循环结构用于重复执行特定的语句块,直到循环终止条件成立或者遇到 break 语句退出循环。在程序设计中经常需要将一些代码重复地执行,使用基本语句顺序执行效率非常低。PHP 提供了 4 种循环结构。

- for 循环：for 循环重复执行语句块,每次重复执行前验证循环条件是否成立,若循环条件成立则循环,否则不再循环。

- do while 循环：do while 语句同样重复执行语句块，在每次结束循环时验证循环条件是否成立，若循环条件成立则继续执行下一次循环，否则不再循环。
- while 循环：while 循环先判断循环条件是否成立，若循环条件成立则循环，否则不循环，甚至有可能一次循环也不执行。
- foreach 循环：foreach 循环用于遍历每一个数组元素。

4.2.1 for 语句

for 循环是 PHP 中较为复杂的循环结构，其功能和用途与 C 语言中的 for 循环类似。for 循环的语法格式如下。

```
for ([表达式 1];[表达式 2];[表达式 3])
{
  语句块
}
```

for 语句循环执行流程如图 4-7 所示。

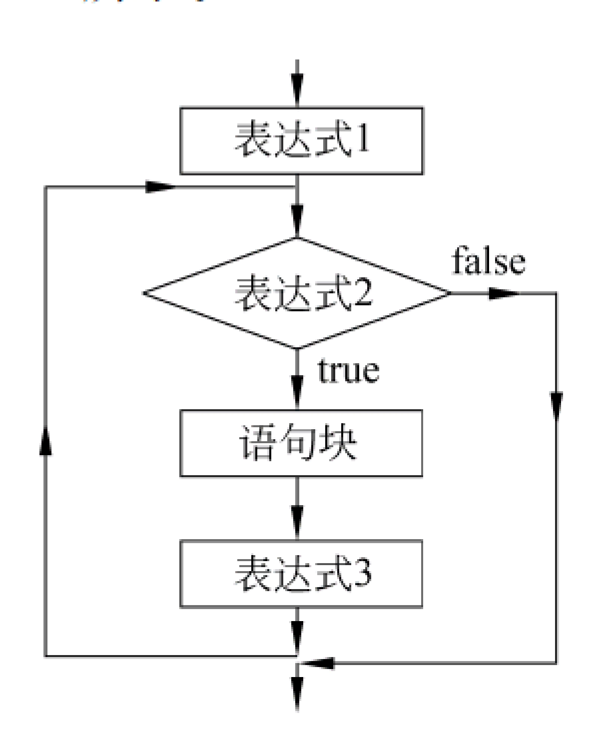


图 4-7 for 语句执行流程

其中，表达式 1 在循环开始前先无条件执行一次。表达式 2 在每次循环之前求值，如果值为 true 则继续循环，值为 false 则结束循环。表达式 3 在每次循环之后执行。每个表达式都可以为空，若表达式 2 为空时，PHP 会认为循环条件成立，循环将无限进行下去，除非有 break 语句强行退出。

说明：

- 表达式 1 无条件首先执行 1 次，而且只执行一次，一般是一些类似变量初始化方面的工作。
- 每次循环都要计算表达式 2 的值：若表达式 2 为 true 则执行语句块，而表达式 3 每次都是在语句块执行完后执行的；若表达式 2 的值为 false 则退出整个 for 循环结构。
- 任何时候碰到 break 语句，都会无条件退出整个 for 循环结构。
- 特别要强调的是：for 循环小括号之后不可以随意加分号，加分号则表示后面的语句块不再属于 for 循环，因为系统把分号(分号可以理解为空语句)理解为 for 循环的语句块。

【示例 7】 求 1＋2＋3＋…＋100 的和。

步骤 1　eg7-1. php 代码如下。

```
<?php
  for($s=0,$i=1; $i<=100; $i++)
     $s +=$i;
  echo $s;
?>
```

步骤 2　eg7-2. php 代码如下。

```
<?php
  $i=1;
  $s=0;
  for( ; $i<=100; ){
     $s +=$i;
     $i++;
  }
  echo $s;
?>
```

步骤 3　eg7-3. php 代码如下。

```
<?php
  $i=1;
  $s=0;
  for( ; ; ){
     if ($i ==101)
         break;
     $s +=$i;
     $i++;
  }
  echo $s;
?>
```

说明：

- 在 eg7-1. php 中表达式 1 用于初始化 $i 和 $s；表达式 2 用于判断是否需要循环；表达式 3 用于修改循环变量。
- 在 eg7-2. php 中初始化在循环之外已提前做好，因此表达式 1 省略不写；表达式 2 用于判断是否需要循环；因为表达式 3 一般都在语句块执行完之后再执行，因此 eg7-2. php 中直接把“$i＋＋”放置在语句块的最后。
- 在 eg7-3. php 中初始化在循环之外，因此无表达式 1；表达式 2 没有，表明循环将无限制进行下去，但是在语句块中有语句“if($i ＝＝ 101) break;”，该语句可以在 $i ＝＝101 时无条件退出 for 循环；表达式 3 没有，直接将“$i＋＋”写在语句块之后。

4.2.2　do...while 语句

do...while 循环在每次循环结束之后再判断循环条件，这种循环至少要循环 1 次。do...while 循环的语法如下。

```
do{
    语句块
}while ( 条件 );
```

其中，如果语句块只有一个语句，则大括号可以不写。while 条件后的小括号之后有分号，这点与 for 语句不相同。do…while 语句执行流程如图 4-8 所示。

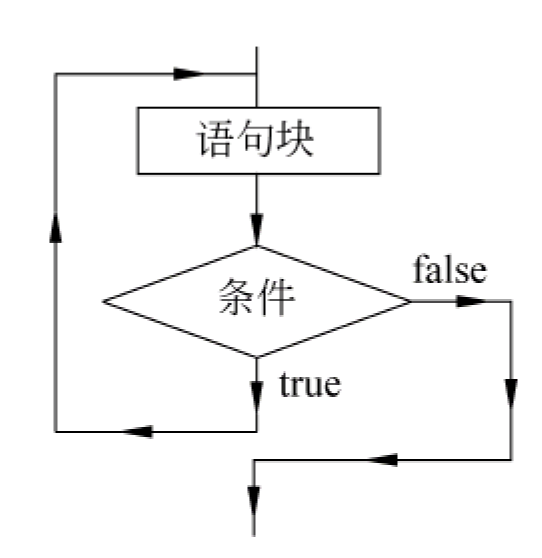


图 4-8　do…while 语句执行流程

说明：

- do…while 循环首先执行语句块，也就是说语句块至少要执行 1 次。
- 语句块执行完毕再来判断条件：若条件为 true，则转到语句块继续循环；若条件为 false，则结束整个 do…while 循环模块。

【示例 8】　使用 do…while 语句求 1+2+3+…+100 的和。

步骤 1　eg8-1.php 代码如下。

```
<?php
  $i =1;
  $s =0;
  do{
    $s =$s +$i;
    $i++;
  }while ($i <=100);
  echo "1+2+3+…+100=$s";
?>
```

步骤 2　eg8-2.php 代码如下。

```
<?php
  $i =0;
  $s =0;
  do{
    $i++;
    $s =$s +$i;
  }while ($i <100);
  echo "1+2+3+…+100=$s";
?>
```

说明：

- 在 eg8-1.php 中 $i 初始值为 1，因此在循环中不要先执行“$i++”；循环执行条件为“$i <= 100”。

- 在 eg8-2. php 中 $i 初始值为 0，因此在循环中可以先执行“$i++”。也可以先执行“$s = $s+1;”。此处是先执行 $i++，为了确保 100 加到 $s 中，我们将循环执行条件设置为“$i < 100”。
- 在 do...while 循环中千万要注意临界点。

4.2.3　while 语句

while 循环比较简单，其基本语法格式如下。

```
while (表达式)
{
    语句块
}
```

与 do...while 循环结构不同的是，while 语句首先要执行表达式，若表达式为 true，则执行循环语句块；否则不执行循环语句块。while 语句执行流程如图 4-9 所示。

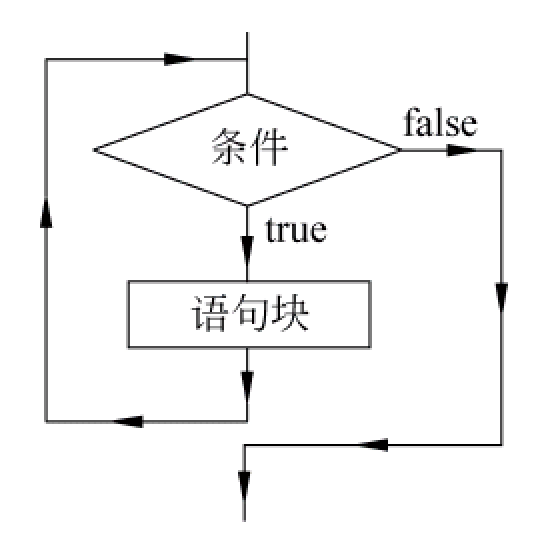


图 4-9　while 语句执行流程

说明：

- while 循环首先判断条件：若为 true 则执行语句块，然后再回到条件判断；若为 false 则结束整个 while 循环模块。
- 该循环有可能 1 次也不循环。
- while 表达式后面的小括号之后不可以随意加分号，加分号后系统会把分号(理解为空语句)当作 while 的语句块。

【示例 9】　使用 while 语句求 1+2+3+…+100 的和。

步骤 1　eg9-1. php 代码如下。

```
<?php
  $i =0;
  $s =0;
  while ($i<100)
  {
    $i++;
    $s =$s +$i;
  }
  echo "1+2+3+4+…+100=$s.";
?>
```

步骤 2　eg9-2. php 代码如下。

```
<?php
  $i =0;
  $s =0;
  while ($i<=100)
  {
    $s =$s +$i;
    $i++;
  }
  echo "1+2+3+4+…+100=$s.";
```

```
?>
```

说明：

- 在 eg9-1.php 中“$i++”在“$s = $s + $i;”前，因此条件设置为“$i<100”。
- 在 eg9-1.php 中“$i++”在“$s = $s + $i;”后，因此条件设置为“$i<=100”。
- 在做这样的题目时一定要注意边界问题，包括初始边界和结束边界。

4.2.4 foreach 语句

foreach 语法结构提供了遍历数组的简单方式。foreach 仅能够应用于数组和对象，如果尝试应用于其他数据类型的变量，或者未初始化的变量，将发出错误信息。有两种语法：

```
foreach (array_expression as $value)
  {语句块}

foreach (array_expression as $key => $value)
  {语句块}
```

第一种格式遍历给定的 array_expression 数组。每次循环中，当前单元的值被赋给 $value 并且数组内部的指针向前移一步（因此下一次循环中将会得到下一个单元）。

第二种格式做同样的事，只除了当前单元的键名也会在每次循环中被赋给变量 $key。

【示例 10】 输出数组元素。

eg10.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $array1 =array('01','name'=>'Liubei','sex'=>'male','English'=>98,'武汉市');
  echo '格式 1:<br />';
  foreach($array1 as $value){
    echo $value."   ";
  }
  echo "<br />";
  echo '格式 2:<br />';
  foreach($array1 as $key=>$value){
    echo $key.'---'.$value."<br />";
  }
?>
```

程序运行结果如图 4-10 所示。

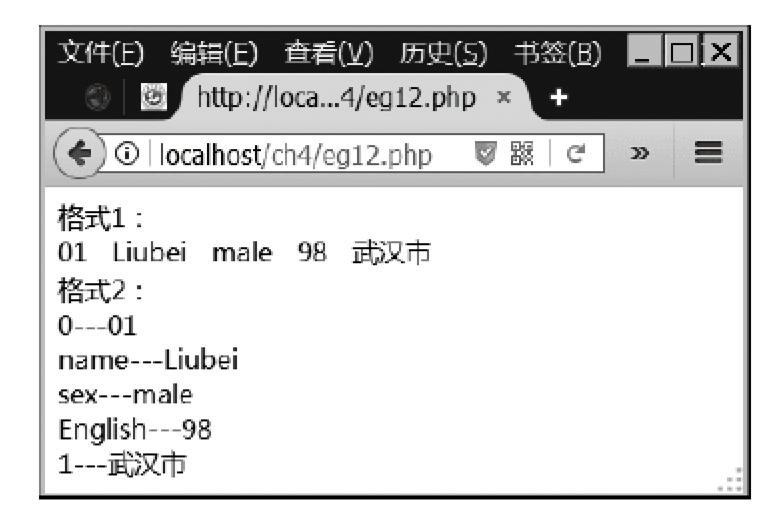


图 4-10 应用 foreach 语句的程序运行结果

说明：

- 格式 1："foreach($ array1 as $ value)"用于访问数组的值。
- 格式 2："foreach($ array1 as $ key=> $ value)"用于访问数组的值和键。

4.2.5　break 语句

break 结束当前 for、foreach、while、do-while 或者 switch 结构的执行。break 可以接受一个可选的数字参数来决定跳出几重循环。

【示例 11】 判断一个数是否是素数。

分析：一个数 n 是素数，则这个数只有两个因子：1 和 n，因此可以循环地在 $2 \sim n-1$ 找 n 的因子，只要找到一个因子即可以判断 n 不是素数，然后立即退出循环，无须再找下一个因子。若循环结束还找不到 n 的因子，则 n 就是素数。根据分析得到 eg11. php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $n=97;
  for($i=2;$i<$n;$i++)
      if ($n % $i ==0)
          break;
  if ($i ==$n)
      echo "{$n}是素数";
  else
      echo "{$n}不是素数!";
?>
```

说明：本题中 for 循环的退出有两种情况：第 1 种是 n 没有因子，if 语句的条件始终不成立，此时正常退出循环，退出后 $ i== $ n。第 2 种情况是 n 有因子，if 语句的条件将成立，此时执行 break 并提前退出循环导致 $ i< $ n，因此可以把 $ i== $ n 看作素数的条件。

4.2.6　continue 语句

continue 在循环结构中用来跳过本次循环中剩余的代码并在条件求值为真时开始执行下一次循环。continue 接受一个可选的数字参数来决定跳过几重循环到循环结尾。默认值是 1，即跳到当前循环末尾。

【示例 12】 输出 100 以内 9 的倍数。

eg12. php 代码如下。

```
<?php
  for($i=1;$i<100;$i++){
      if ($i % 9 <>0)
          continue;
      echo $i."   ";
  }
?>
```

说明：if 语句判断 $ i 是否是 9 的倍数：若不是 9 的倍数(if 条件成立)则执行 continue

语句,有意漏掉语句“echo $i."   ";”,确保不输出 9 的倍数之外的数;若是 9 的倍数,则 if 条件不成立,顺序向下执行语句“echo $i."   ";”,输出 9 的倍数。

4.3 分支和循环的嵌套

本节将讲述各种流程控制的综合举例,包括各种分支结构、各种循环结构以及分支结构和循环结构的嵌套。

4.3.1 分支语句嵌套

分支结构中又有分支结构,就称为分支结构的嵌套。

【示例 13】 根据分数给出学生成绩的等级。

步骤 1 eg13-1.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $score = 95;
  $grade = '';
  if ($score >=70){
    if ($score >=85)
        $grade = '优秀';
    else
        $grade = '良好';
  }else{
    if ($score >=60)
        $grade = '及格';
    else
        $grade = '不及格';
  }
  echo $grade;
?>
```

步骤 2 eg13-2.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $score = 95;
  $grade = '';
  if ($score >=85)
      $grade = '优秀';
  else {
      if ($score >=70)
          $grade = '良好';
      else {
          if ($score >=60)
            $grade = '及格';
```

```
        else
            $grade = '不及格';
    }
  }
  echo $grade;
?>
```

说明：相比较而言，eg13-1.php 更加优化，因为对分数比较的次数比较少。在现实编程中还要考虑分数的分布情况，尽量让分数段多的那些分数比较的次数更少。如果是对多个分数进行等级输出，可以考虑循环加分支结构嵌套。

4.3.2　循环语句嵌套

在循环语句中使用循环语句在程序设计中非常普遍。

【示例 14】　输出 100 以内的素数。

eg14.php 代码如下。

```
<?php
  $count = 0;
  for($n=2;$n<=100;$n++){
     $flag = true;                        //假设$n是素数
     for ($i=2; $i<$n;$i++)
         if ($n % $i == 0){
            $flag = false;                //$n有因子，$n不是素数
            break;                        //无须继续循环再找，因此退出内循环
         }
     if ($flag){                          //若$n是素数，则输出
         $count ++;                       //$count比较素数的个数
         echo $n."   ";
         if ($count % 5 == 0)             //每行输出 5个素数
            echo '<br />';
     }
  }
?>
```

说明：

- 内循环用于判断 $n 是否是素数，内循环结束时，若是素数则输出。
- 外循环用于让 $n 从 2 到 100 循环，因为 1 不是素数，无须判断 1。
- $count 则用于计数，每行输出 5 个素数，$count 是 5 的倍数时则换行。

【示例 15】　输出如图 4-11 所示的图形。

```
                1
              1 2 1
            1 2 3 2 1
          1 2 3 4 3 2 1
        1 2 3 4 5 4 3 2 1
      1 2 3 4 5 6 5 4 3 2 1
    1 2 3 4 5 6 7 6 5 4 3 2 1
  1 2 3 4 5 6 7 8 7 6 5 4 3 2 1
1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1
```

图 4-11　打印数字

分析：要想打印如图 4-11 所示的图形，可以考虑使用循环来打印 9 行“for($i=1; $i<10; $i++)”，其中 $i 控制行数。而每一行又分四部分来打印：第一部分是每行前面的空格；第二部分是递增的数字；第三部分是递减的数字；第四部分是换行符。第一部分的空格数随着行数的增加而减少，因此可以考虑写成“for($j=1; $j<= $x- $i; $j++) echo ' ';”的形式，$x 越大则前面

的空格数就越多;第二部分是递增的数字,使用语句“for($j=1; $j<= $i; $j++) echo $j.' ';”来打印;第三部分是递减的数字,使用语句“for($j=1; $j< $i; $j++) echo $i- $j.' ';”来打印;第四部分是换行。根据分析而得到的代码如下。

eg15 代码如下。

```
<?php
  for($i=1;$i<10;$i++){              //合计有 9行
    for($j=1;$j<=10-$i;$j++)         //每行前面的空格数递减
       echo '  ';
    for($j=1;$j<=$i;$j++)            //数字递增
       echo $j.' ';
    for($j=1;$j<$i;$j++)             //数字递减
       echo $i-$j.' ';
    echo '<br />';                   //换行
  }
?>
```

4.3.3 混合语句嵌套

嵌套不仅用于分支语句、循环语句,还可以用于分支语句和循环语句之间。

【示例 16】 输出一个大于 10000000 的素数。

分析:要输出大于 10000000 的素数,则偶数是可以排除的,因此可以考虑从 10000001 开始循环,每次循环变量加 2。

eg16.php 代码如下。

```
<?php
  for($n=10000001;true;$n=$n+2){      //不考虑偶数,循环次数未知
    $flag =true;                      //假设$n是素数
    for($i=2;$i<sqrt($n);$i++){       //查找$n的因子时缩小检测范围
      if ($n %$i==0){                 //$n有因子,$n不是素数
           $flag =0;                  //$n不是素数
           continue 2;                //若$n有因子,则直接进入下一次外循环并考虑下一个
                                        $n
      }
    }
    if ($flag){                       //内循环正常结束,且$flag=true,说明$n是素数
         echo $n;
         break;                       //找到素数时即可退出
    }
  }
?>
```

说明:关于 n 的因子说明如下。若 n 有因子,则成对出现在[1,$\sqrt{n}$]和[$\sqrt{n}$,n]之间。比如 100 的因子出现在[1..10]和[10..100]之间的因子分别为 1、2、4、5、10 和 100、50、25、20、10。因此在检测 n 的因子时把范围缩小到[2,$\sqrt{n}$]之间,从而加快了程序运行的速度。

4.4　综合案例——验证哥德巴赫猜想

18 世纪上半叶，德国数学家哥德巴赫偶尔发现，每个不小于 6 的偶数都是两个素数之和。例如 6＝3＋3，24＝11＋13。他经过长时间的验算之后，试图证明自己这一发现，然而屡试屡败。1742 年，毫无办法的哥德巴赫写信求教于当时世界上最有权威的瑞士数学家欧拉，提出了自己的猜想，并问这是否是一个定理。欧拉很快回信说，这个猜想肯定是定理，但我无法证明它。

本综合案例将给定一个有限大小的偶数并尝试验证哥德巴赫猜想（并非数学上的理论证明）。

首先给定 $n，例如 $n＝1000；再给定 $a ＝ 3（不考虑偶数，大的偶数肯定不是素数），$b ＝ $n－ $a，此时 $n＝ $a＋ $b 是成立的，若 $a 和 $b 都是素数，则问题得到了验证。

若 $a 和 $b 中只要有一个不是素数，则需要考虑其他组合。此时修改 $a 和 $b 的组合，即 $a ＝ $a ＋ 2，$b ＝ $n － $a，仍然保持 $n ＝ $a ＋ $b，若此时 $a 和 $b 都是素数，则问题得到验证。若 $a 和 $b 中只要有一个不是素数，则需要继续换一个组合，为了不漏掉任何一个组合，$b 每次都加 2，即 $a ＝ $a ＋2，$b ＝ $n－ $a。

继续周而复始地进行相同的验证操作，直到 $a>＝ $b 为止，因为若 $a>＝ $b 后再继续验证也是重复验证，因为 $a 和 $b 是同等关系。

步骤 1　根据以上分析，eg17.php 的代码如下。

```
<?php
  //判断$n是否是素数
  function is_prime($n){
    $flag=true;                    //假设$n是素数
    for($i=2;$i<=sqrt($n);$i++)
      if ($n%$i==0){
          $flag=0;                 //找到因子无须继续判断
          break;
      }
    return $flag;
  }
  $n=10000000;
  $a=1;
  $b=$n-$a;
  do
  {
    $a=$a+2;
    $b=$n-$a;
    //如果$a和$b都是素数,则输出结果,不再循环
    if (is_prime($a) && is_prime($b)){
       echo "$n=$a+$b";
       break;
    }
  }
```

```
    //若$a>$b,则无须继续循环,重复的组合省略
    while ($a<=$b);
?>
```

程序运行结果如下。

```
10000000 =29 +9999971
```

步骤 2 如果想输出符合哥德巴赫猜想的全部组合,则代码可以修改为 eg17-2.php。

```
<?php
  //判断$n是否是素数
  function is_prime($n){
    $flag=true;                    //假设$n是素数
    for($i =2 ; $i<=sqrt($n); $i++)
      if ($n %$i==0){
          $flag=0;
          break;
      }
    return $flag;
  }
  $n=200;
  $a=1;
  $b=$n-$a;
  do
  {
    $a=$a+2;
    $b=$n-$a;
    //如果$a和$b都是素数,则输出结果,不再循环
    if (is_prime($a) && is_prime($b)){
       echo "$n=$a+$b<br />";      //不跳出循环,输出全部组合
    }
  }
  //若$a>$b,则无须继续循环,重复的组合省略
  while ($a<=$b);
?>
```

程序运行结果如下。

```
200 =3 +197
200 =7 +193
200 =19 +181
200 =37 +163
200 =43 +157
200 =61 +139
200 =73 +127
200 =97 +103
```

4.5 习　　题

一、填空题

1. 除了顺序结构外,还有选择语句、循环语句和________。

2. 选择语句有 if 语句、if...else 语句、________语句。
3. 跳转语句有 break 语句和________语句。
4. ________循环先执行语句块再进行循环条件判断。
5. 用于数组的循环语句是________。
6. if 或 if...else 语句的嵌套遵循________。

二、选择题

1. 下列选项中不属于嵌套的是________。

A.
```
for()
{ if () {} }
```

B.
```
for ( )
{ for ( )

{ }
}
```

C.
```
switch()
{
  case
  break;
}
```

D.
```
if()
{ if ( ) { } }
```

2. 下面程序运行后，输出________个 4。

```
<?php
  for($a=0;$a<6;$a++)
    for($i=0;$i<$a;$i++)
  echo $a."<br />";
?>
```

A. 3　　B. 4　　C. 5　　D. 6

3. 下面的程序运行的结果是________。

```
<?php
  for($a=0;$a<10;$a++){
    if ($a%4==0)
      continue;
    echo $a;
  }
?>
```

A. 1235679　　B. 123567910　　C. 48　　D. 都不是

4. 下面的程序运行的结果是________。

```
<?php
  for($a=0;$a<10;$a++){
    if ($a%4==0)
  break;
  echo $a;
  }
?>
```

A. 没有输出　B. 123　C. 0　D. 都不是

5. 下面的程序运行的结果是________。

```
<?php
  $arr =array(1,3,5,7,'name'=>'Liubei');
  foreach($arr as $key =>$a)
    echo $key;
?>
```

A. 0123name　B. 0123　C. 1357name　D. 1357Liubei

6. 下面的程序运行的结果是________。

```
<?php
  $mon =5;
  $year =2017;
  switch($mon){
    case 1:
    case 3:
    case 5:
    case 7:
    case 8:$days =31;
    case 4:
    case 6:
    case 9:
    case 11:$days =30;
    case 2: $days =28;
  }
  echo $days
?>
```

A. 28　B. 29　C. 30　D. 31

7. 下面的程序运行的结果是________。

```
<?php
  $n =37;
  for($i=2;$i<$n;$i++)
    if ($n %$i ==0)
      break;
  echo $i."  ";
  $n =36;
  for($i=2;$i<$n;$i++)
    if ($n %$i ==0)
      break;
  echo $i;
?>
```

A. 37　36　B. 37　2　C. 没有输出　D. 以上都不对

8. 下面的程序运行的结果是________。

```
<?php
  $a =34;
  $b =true;
```

```
    $c=8;
    if (($a==$b) && ($c=10))
      echo $c;
?>
```

A. 8　　　B. 10　　　C. 无输出　　　D. 都不对

三、程序设计题

1. 使用循环语句打印如下图案。

```
*********
 ********
  *******
   ******
    *****
     ****
      ***
       **
        *
```

2. 编写程序，判断一个数是否为素数。
3. 编写程序，输出一个数的全部因子。
4. 给定 n，编程求 $1+(1+2)+(1+2+3)+(1+2+3+4)\cdots+(1+2+3+\cdots+n)$。

第5章　PHP函数

知识点：

- 函数的分类
- PHP 变量处理函数
- PHP 数学函数
- PHP 日期时间函数
- 自定义函数的创建和调用
- 自定义函数参数的传递方式
- 自定义函数的返回值
- 嵌套函数和递归函数

本章导读：

对于程序设计语言来说，总会出现一些功能重复的代码，例如连接数据库代码。对于一个比较大的程序而言代码的重复出现非常不利于代码的书写、使用和阅读，也不利于后期的维护。于是 PHP 把这些具有一定功能的代码以函数的形式封装起来。在程序的不同位置需要使用到这些代码块时，只需要使用该代码块的名字(函数名)即可反复多次调用，这样大大地提高了工作效率。如果代码块出现问题，修改起来也非常方便。

本章将介绍 PHP 变量处理函数、PHP 字符串处理函数、日期时间函数、数学函数、自定义函数、参数传递、递归和嵌套函数等内容。字符串函数和数组函数在相关章节介绍。

5.1　PHP函数概述

在 PHP 中为了实现一定的功能，将一些代码写入一个命名的代码块中，在必要的时候可以多次调用它，这个代码块就是函数。

函数是完成一定功能的代码集合，在 PHP 中函数可以分为两类：一类是系统预定义的函数；另一类是用户自定义函数，用户需要先定义才能使用。很多系统函数都已经编译到安装发行包中，可以直接使用。有些函数没有编译到安装发行包中，需要使用 include()或 require()包含进来才可以使用。

系统预定义的函数又称为内置函数或系统函数。系统函数分为两部分。一部分是核心函数，例如字符串函数和变量函数，在各个版本的 PHP 中会随着 PHP 的安装而安装。还有一些函数则需要和特定的 PHP 模块一起安装。

用户自定义函数是提供独立明确任务的代码块，主要目的是提高程序的效率。

不管哪一类函数，都需要通过函数名和参数来调用。即使没有参数，函数的括号也不能少。

5.2　变量处理函数

变量处理函数用于对变量进行处理，例如判断变量是否存在、变量类型的转换、变量的销毁等。前面一些章节已经讲述了一些变量处理函数。表 5-1 对常用的 PHP 变量处理函数进行了总结说明。

表 5-1　PHP 常用的变量处理函数

函 数 名 称	说　　明
doubleval()、floatval()	把变量转变成双精度浮点型
empty()	判断变量是否为空
gettype()	获得变量的类型
intval()	把变量转换成整型
is_array()	判断变量是否为数组类型
is_double()	判断变量是否为双精度类型
is_float()	判断变量是否为浮点型
is_int()、is_integer()、is_long()	判断变量是否为整型
is_object()	判断变量是否为对象类型
is_real()	判断变量是否为实型
is_string()	判断变量是否为字符串类型
isset()	判断变量是否存在
settype()	设置变量类型
strval()	将变量转换为字符串类型
unset()	销毁变量

下面分别列出这些函数的格式，有些函数在前面章节中已经讲述，此处不再赘述。

1. empty()函数

```
bool empty ( mixed $var )
```

如果 var 是非空或非零的值，则 empty()返回 false。换句话说，""、0、"0"、NULL、false、array()、“var $var;”以及没有任何属性的对象都将被认为是空的，如果 var 为空，则返回 true。

2. isset()函数

```
bool isset ( mixed $var [, mixed $ ...] )
```

检测变量是否设置，并且不是 NULL。如果已经使用 unset()释放了一个变量之后，它将不再是 isset()。若使用 isset()测试一个被设置成 NULL 的变量，将返回 false。

3. unset()函数

```
void unset ( mixed $var [, mixed $ ...] )
```

unset()销毁指定的变量。unset()在函数中的行为会依赖于想要销毁的变量的类型而有所不同。如果在函数中有一个全局变量,则只是局部变量被销毁,而在调用环境中的全局变量将保持与调用 unset()之前一样的值。

【示例 1】 PHP 与变量有关的函数举例。

eg1.php 代码如下。

```
<?php
  $var  =  0;
  //结果为 true,因为$var 为空
  if (empty($var)) {
      echo '$var is either 0 or not set at all.<br />';
  }
  //结果为 false,因为$var 已设置
  if (!isset($var)) {
      echo '$var is not set at all';
  }
  $var2 = 'Hello';
  if (isset($var2))              //结果为 true,因$var2 存在且不为 NULL
      echo $var2.'<br />';       //有输出
  unset($var,$var2);             //销毁$var 和$var2
  if (isset($var2))              //结果为 false,因$var2 不存在
      echo $var2.'<br />';       //不输出
?>
```

5.3 数学函数

数学函数是 PHP 标准数据库中的一个类别,这些函数可以处理计算机中整型和实型范围内的值。表 5-2 给出了 PHP 常用的数学函数。

表 5-2 PHP 中常用的数学函数

函数名称	说　明	函数名称	说　明
abs()	求绝对值	rand()和 mt_rand()	产生随机数
sin()	求正弦	round()	四舍五入取整
asin()	求反正弦	sqrt()	求平方根
cos()	求余弦	log()	求自然对数
acos()	求反余弦	log10()	求以 10 为底的对数
tan()	求正切	pi()	求圆周率
atan()	求反正切	rad2deg()	弧度转换为角度
ceil()	进一法取整	deg2rad()	角度转化为弧度
floor()	舍去法取整	pow()	求指数
max()	求最大值	exp()	求 e 的指数
min()	求最小值		

5.3.1　三角函数

1. 求正弦函数 sin()和反正弦函数 asin()

(1) 求正弦函数的格式为：

```
float sin ( float $arg )
```

sin() 返回参数 arg 的正弦值。参数 arg 的单位为弧度。

(2) 求反正弦函数的格式为：

```
float asin ( float $arg )
```

返回 arg 的反正弦值，单位是弧度。asin() 是 sin() 的反函数，它的意思是在 asin() 范围里的每个值都是 a==sin(asin(a)) 。

2. 求余弦函数 cos()和反余弦函数 acos()

(1) 求余弦函数 cos()格式为：

```
float cos ( float $arg )
```

返回参数 arg 的余弦值。参数 arg 的单位为弧度。

(2) 求反余弦函数 acos()格式为：

```
float acos ( float $arg )
```

返回 arg 的反余弦值，单位是弧度。acos() 是 cos() 的反函数，它的意思是在 acos() 范围里的每个值都是 a==cos(acos(a)) 。

3. 求正切函数 tan()和反正切函数 atan()

(1) 求正切函数 tan()格式为：

```
float tan ( float $arg )
```

tan()返回参数 arg 的正切值。参数 arg 的单位为弧度。

(2) 求反正切函数 atan()格式为：

```
float atan (float $arg )
```

返回 arg 的反正切值，单位是弧度。atan()是 tan()的反函数，它的意思是在 atan()范围里的每个值都是 a==tan(atan(a))。

4. 弧度转换为角度函数和角度转换为弧度函数

(1) 弧度转换为角度函数 rad2deg()格式为：

```
float rad2deg ( float $number )
```

本函数将 number 从弧度转换为角度。

(2) 角度化成弧度函数 deg2rad()的格式为：

```
float deg2rad ( float $number )
```

本函数把 number 从角度转换成弧度。

5. 圆周率函数

圆周率 pi()函数的格式为：

```
float pi ( void )
```

返回圆周率的近似值。返回值的 float 精度是由 php.ini 中的 precision 指令确定。默认值是 14。也可以使用 M_PI 常量，该常量产生与 pi()完全相同的结果。

【示例 2】 PHP 绝对值函数和三角函数举例。

eg2.php 代码如下。

```
<?php
  $alpha = 90;
  $b = sin($alpha * pi()/180);          //90°的正弦
  echo $b.'<br />';                     //输出 1
  $x = 0.5;
  $beita= asin($x);                     //0.5 的反正弦
  echo $beita.'<br />';                 //输出 0.5235987755983，即 pi()/6
  echo rad2deg($beita).'<br />';        //输出 30，pi()/6 化成度为 30°
  $c = cos($alpha * pi()/2);            //90°的余弦
  echo $c.'<br />';                     //输出 -1
  $delta = acos(0.5);                   //0.5 的反余弦
  echo $delta.'<br />';                 //输出 1.0471975511966，即 pi()/3
  $d = tan(45 * pi()/180);              //45°的正切
  echo $d.'<br />';                     //输出 1，45°的正切
  $gama = rad2deg(atan(1));             //1 的反正切
  echo $gama.'<br />';                  //输出 45(°)
  $e = atan(1);                         //1 的反正切
  echo $e.'<br />';                     //输出 0.78539816339745，即 pi()/4
  $e = pi() /4;
  $xita = rad2deg($e);
  echo $xita;                           //输出 45°，pi()/4 化成度就是 45°
?>
```

5.3.2 指数和对数函数

1. 指数函数

(1) 指数函数 pow()的格式为：

```
number pow ( number $base, number $exp )
```

返回 base 的 exp 次方的幂。如果可能，本函数会返回 integer 型数据。

(2) 以 e 为底的指数函数 exp()的格式为：

```
float exp ( float $arg )
```

返回 e 的 arg 次方值。

2. 对数函数

(1) 以 10 为底的对数函数的格式为：

```
float log10 ( float $arg )
```

返回参数 arg 以 10 为底的对数。

（2）以 e 为底的对数函数的格式为：

```
float log ( float $arg [, float $base = M_E ] )
```

如果指定了可选的参数 base，log()返回 logbase arg，否则 log()返回参数 arg 的自然对数。

【示例 3】 PHP 中的对数函数和指数函数应用举例。

eg3.php 代码如下。

```
<?php
  echo pow(2,5).'<br />';                //输出 2^5，即 32
  echo exp(5).'<br />';                  //输出 e^5 的值，即 148.41315910258
  echo log(exp(5)).'<br />';             //输出 log_e e^5 的值，即 5
  echo log(10000,10).'<br />';           //输出 log_10 10000 的值，即 4
  echo log10(pow(10,6)).'<br />';        //输出 log_10 10^6 的值，即 6
?>
```

说明：log()函数有两种格式。第 1 种格式只有一个参数，表示的含义是以 e 为底的对数；第 2 种格式带两个参数，表示的含义是以第 2 个参数为底数，第 1 个参数为真数的对数。

5.3.3　最大函数及最小函数

（1）最大函数 max()的格式为：

```
mixed max ( array $values )
```

或

```
mixed max ( mixed $value1, mixed $value2 [, mixed $ ... ] )
```

如果仅有一个参数且为数组，max()返回该数组中最大的值。如果第一个参数是整数、字符串或浮点数，则至少需要两个参数，而 max()会返回这些值中最大的一个。可以比较无限多个值。

（2）最小函数 min()的格式为：

```
mixed min ( array $values )
```

或

```
mixed min ( mixed $value1, mixed $value2 [, mixed $ ... ] )
```

如果仅有一个参数且为数组，min()返回该数组中最小的值。如果第一个参数是整数、字符串或浮点数，则至少需要两个参数，而 min()会返回这些值中最小的一个。可以比较无限多个值。

【示例 4】 PHP 中最大、最小和平方根函数应用举例。

eg4.php 代码如下。

```
<?php
  $array1 =array(19,'liubei'=>23,'machao'=>17,24);
```

```
  $a =max($array1);
  $b =max(1,3,5,-2);
  $c =min(true,false);
  echo $a.'<br />';                 //24
  echo $b.'<br />';                 //5
  var_dump($c);                     //bool(false)
?>
```

5.3.4 取整函数

(1) 四舍五入取整函数 round()的格式为:

```
float round ( float $val [, int $precision =0 [, int $mode =PHP_ROUND_HALF_UP ]] )
```

返回值将根据指定精度 precision(十进制小数点后数字的数目)进行四舍五入。precision 也可以是负数或零(默认值)。

(2) 进一法取整函数 ceil()的格式为:

```
float ceil ( float $value )
```

返回值为不小于 value 的下一个整数。value 如果有小数部分,则进一位。

【示例 5】 PHP 中取整根函数应用举例。

eg5.php 代码如下。

```
<?php
  $a =round(4.66);
  echo $a.'<br />';                 //输出 5,四舍五入
  $b =round(-4.6);
  echo $b.'<br />';                 //输出-1,四舍五入
  $c =round(4.51267,3);
  echo $c.'<br />';                 //输出 4.513,保留三位小数,第 4 位四舍五入
  $d =ceil(4.9);
  echo $d.'<br />';                 //输出 5,变大则取整
  $e =ceil(-4.9);
  echo $e.'<br />';                 //输出-4,变大则取整
?>
```

5.3.5 其他函数

(1) 绝对值函数 abs()的格式为:

```
number abs ( mixed $number )
```

返回 number 参数的绝对值。

(2) 算术平方根函数 sqrt()的格式为:

```
float sqrt ( float $arg )
```

返回 arg 的平方根。

(3) 随机函数 rand()的格式为:

```
int rand ( void )
```

或

```
int rand ( int $min, int $max )
```

如果没有提供可选参数 min 和 max，rand()返回 0～getrandmax()的伪随机整数。例如，想要 5～15(包括 5 和 15)的随机数，用 rand(5,15)。

【示例 6】 PHP 中绝对值、平方根、随机函数应用举例。

eg6.php 代码如下。

```
<?php
  echo abs(-4.6).'<br />';            //输出 4.6
  echo sqrt(81).'<br />';             //输出 9
  echo rand().'<br />';               //输出 0 ~getrandmax() 的伪随机整数
  echo rand(1,100).'<br />';          //输出 1~100 的伪随机数
  echo chr(rand(97,97+25));           //随机输出一个小写的英文字符
?>
```

5.4　日期和时间函数

在 PHP 中存在一类日期时间函数，使用这些函数可以得到 PHP 所运行服务器的当前日期及时间，并且可以使用不同的格式将这些日期时间显示出来。PHP 中常见的日期时间函数见表 5-3。

表 5-3　PHP 中常用的日期和时间函数

函数名称	说　　明
checkdate	检测日期是否合法
getdate	以数组形式返回当前的日期和时间
date	将整数时间转变为字符串格式
strtotime	将英文字符串日期时间转换为 UNIX 时间标签
microtime	将 UNIX 时间标签格式转化为适用于当前环境的日期字符串
gmdate	将 UNIX 时间标签格式化成日期字符串
time	返回当前的时间戳

5.4.1　checkdate()函数和 getdate()函数

1. checkdate()函数

checkdate()函数常常用于计算日期或者将日期保存到数据库之前，可以使用该函数检测该日期是否合法。checkdate()函数的格式为：

```
bool checkdate ( int $month, int $day, int $year )
```

检查由参数构成的日期的合法性。如果每个参数都已经正确定义，则会被认为是有效

的。若日期有效则返回 true,否则返回 false。

2. getdate()函数

getdate()函数用于获取当前的日期和时间。一般情况下可以使用该函数获得一系列离散的日期和时间值。getdate()函数的格式为:

```
array getdate ([ int $timestamp = time() ] )
```

该函数返回一个根据 timestamp 得出的包含有日期信息的关联数组 array。如果没有给出时间戳,则认为是本地当前时间。参数 timestamp 是可选的,该参数是一个 integer 类型的 UNIX 时间戳,如未指定,参数值默认为本地当前时间。也就是说,其值默认为 time()的返回值。表 5-4 给出了 getdate()函数返回数组中的键名和关联值。

表 5-4 getdate()函数返回数组中的键名和关联值

键　名	说　　明	返回值例子
seconds	秒的数字表示	0～59
minutes	分钟的数字表示	0～59
hours	小时的数字表示	0～23
mday	月份中第几天的数字表示	1～31
wday	星期中第几天的数字表示	0 (周日)～6 (周六)
mon	月份的数字表示	1～12
year	4 位数字表示的完整年份	比如 1999 或 2003
yday	一年中第几天的数字表示	0～365
weekday	星期几的完整文本表示	Sunday～Saturday
month	月份的完整文本表示,比如 1 月就是 January	January～December
0	自从 UNIX 纪元开始至今的秒数	范围为－2147483648～2147483647

【示例 7】 checkdate()函数和 getdate()函数应用举例。

eg7. php 代码如下。

```
<?php
  header('content-type:text/html;charset=utf-8');
  echo checkdate(3,31,2017)?'合法':'非法';          //输出为"合法"
  echo '<br />';
  echo checkdate(2,29,2017)?'合法':'非法';          //输出为"非法"
  echo '<br />';
  $now=getdate();
  echo '现在是'.$now['year'].'年'.$now['mon'].'月'.$now['mday'].'<br />';
  //输出:现在是 2017 年 2 月 16 日
  echo '今天是'.$now['weekday'].'<br />';           //输出:今天是 Thursday
  echo '今天是星期'.$now['wday'].'<br />';           //输出:今天是星期 4
  echo '今天是今年的第'.$now['yday'].'天<br />';  //输出:今天是今年的第 46 天
  echo '本月的第'.$now['mday'].'天<br />';           //输出:本月的第 16 天
  echo '现在是'.$now['hours'].'时'.$now['minutes'].'分'.$now['seconds'].'秒<br />';
  //输出:现在是 11 时 5 分 19 秒
?>
```

说明：若语句“$now = getdate();”出现 warning 信息，则需要在 php.ini 里加上 date.timezone 项，并设置 date.timezone = "Asia/Shanghai"，然后重新启动 Apache 即可。

5.4.2　date()函数

date()函数可以格式化一个日期和时间，在网页中经常用到。date()函数的语法格式为：

```
string date ( string $format [, int $timestamp ] )
```

返回将整数 timestamp 按照给定的格式字串而产生的字符串。如果没有给出时间戳，则使用本地当前时间。换句话说，timestamp 是可选的，默认值为 time()。表 5-5 给出了 $format 参数格式化说明。

表 5-5　$format 参数格式化说明

键名	说　明	返回值例子
d	月份中的第几天，是有前导零的 2 位数字	01～31
D	星期中的第几天，文本表示，用 3 个字母	Mon～Sun
j	月份中的第几天，没有前导零	1～31
l	L 的小写，表示星期几，完整的文本格式	Sunday～Saturday
N	ISO—8601 格式数字表示的星期中的第几天(PHP 5.1.0 中新加)	1(表示星期一)～7(表示星期天)
S	每月天数后面的英文后缀，用 2 个字符	st、nd、rd 或者 th。可以和 j 一起用
w	星期中的第几天，用数字表示	0(表示星期天)～6(表示星期六)
z	年份中的第几天	0～365
星期		
W	ISO—8601 格式年份中的第几周，每周从星期一开始(PHP 4.1.0 新加的)	例如 42(当年的第 42 周)
月		
F	月份，用完整的文本格式，例如 January 或者 March	January～December
m	数字表示的月份，有前导零	01～12
M	用三个字母的缩写表示的月份	Jan～Dec
n	数字表示的月份，没有前导零	1～12
t	给定月份所应有的天数	28～31
年		
L	判断是否为闰年	如果是闰年，则为 1，否则为 0
o	ISO—8601 格式的年份数字。这和 Y 的值相同，只有当 ISO 的星期数(W)属于前一年或下一年，则用那一年(PHP 5.1.0 中新加)	例如 1999 或 2003
Y	4 位数字完整表示的年份	例如 1999 或 2003
y	2 位数字表示的年份	例如 99 或 03
时间		
a	小写的上午和下午值	am 或 pm
A	大写的上午和下午值	AM 或 PM

续表

键名	说　　明	返回值例子
B	Swatch Internet 标准时	000～999
g	表示小时，12 小时格式，没有前导零	1～12
G	表示小时，24 小时格式，没有前导零	0～23
h	表示小时，12 小时格式，有前导零	01～12
H	表示小时，24 小时格式，有前导零	00～23
i	有前导零的分钟数	00～59
s	表示秒数，有前导零	00～59
	时区	
e	时区标识（PHP 5.1.0 中新加）	例如，UTC、GMT、Atlantic/Azores
I	判断是否为夏令时	如果是夏令时，则为 1，否则为 0
O	与格林尼治时间相差的小时数	例如，＋0200
P	与格林尼治时间（GMT）的差别，小时和分钟之间有冒号分隔（PHP 5.1.3 中新加）	例如，＋02:00
T	本机所在的时区	例如，EST、MDT（在 Windows 下为完整文本格式，例如 Eastern Standard Time，中文版会显示“中国标准时间”）
Z	时差偏移量的秒数。UTC 西边的时区偏移量总是为负的，UTC 东边的时区偏移量总是为正的	－43200～43200
	完整的日期/时间	
c	ISO—8601 格式的日期（PHP 5 中新加）	2004-02-12T15:19:21＋00:00
r	RFC 822 格式的日期	例如，Thu，21 Dec 2000 16：01：07 ＋0200
U	从 UNIX 纪元（January 1 1970 00:00:00 GMT）开始至今的秒数	参见 time()

【示例 8】 date()函数应用举例。

eg8.php 代码如下。

```
<?php
  date_default_timezone_set ( 'UTC' );            //设定要用的默认时区。自 PHP 5.1 开始可用
  //假如现在是 2017 年 2 月 16 日星期四
  echo  date ( "l" ).'<br />';
  //输出类似：Thursday
  echo  date ( 'l dS \of F Y h:i:s A' ).'<br />';
  //输出类似：Thursday 16th of February 2017 07:29:35 AM
  $today  =  date ( "F j, Y, g:i a" );                //如 February 16, 2017, 7:43 am
  $today  =  date ( "m.d.y" );                        //如 02.16.17
  $today  =  date ( "j, n, Y" );                      //如 16, 2, 2017
  $today  =  date ( "Ymd" );                          //如 20170216
  $today  =  date ( 'h-i-s, j-m-y, it is w Day z ' );
//07-43-27, 16-02-17, 4328 4327 4 Thuaml7 46
  $today  =  date ( '\I\t \i\s \t\h\e jS \d\a\y.' );  //如 It is the 16th day.
  $today  =  date ( "D M j G:i:s T Y" );              //如 Thu Feb 16 7:43:27 UTC 2017
```

```
  $today  =  date ( 'H:m:s \m \i\s\ \m\o\n\t\h' );           //如 07:02:27 m is month
  $today  =  date ( "H:i:s" );                               //如 07:43:27
?>
```

5.4.3　time()函数

time()函数返回当前的 UNIX 时间戳，即返回自从 UNIX 纪元（格林尼治时间 1970 年 1 月 1 日 00:00:00）到当前时间的秒数。time()函数的格式为：

```
int time ( void )
```

返回自从 UNIX 纪元（格林尼治时间 1970 年 1 月 1 日 00:00:00）到当前时间的秒数。

使用 time()函数再配合 date()函数可以获得一些日期时间信息。例如，要获取明天的日期，则：

```
$time=time();
$date2 =date('Y-m-d',$time +24 * 3600);
```

【示例 9】　time()函数应用举例。

eg9.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $time =time();
  $date =date('Y-m-d',$time);
  echo $date.'<br />';                                     //输出为 2017-02-16
  $date2 =date('Y-m-d',$time +24 * 3600);                  //明天
  echo $date2.'<br />';                                    //输出为 2017-02-17
  $a =strtotime("$date2");                                 //把明天转为时间戳
  echo date('Y-m-d;星期 N',$a);                            //2017-02-17;星期 5
?>
```

5.4.4　strtotime()函数

strtotime()函数可以将任何英文文本的日期时间描述解析为 UNIX 时间戳。strtotime()函数的语法格式为：

```
int strtotime ( string $time [, int $now =time() ] )
```

本函数预期接受一个包含美国英语日期格式的字符串并尝试将其解析为 UNIX 时间戳（自 January 1 1970 00:00:00 GMT 起的秒数），其值相对于 now 参数给出的时间。如果没有提供此参数，则用系统当前时间。time 是日期时间字符串，time 要求是正确的日期时间格式，now 是用来计算返回的时间戳，默认为现在。返回值是整型时间戳。

【示例 10】　strtotime()函数应用举例。

eg10.php 代码如下。

```
<?php
  echo strtotime('now').'<br />';                          //输出现在的时间戳
  echo strtotime('23 September 2017').'<br />';            //输出 2017-9-23 的时间戳
```

```
    echo strtotime('last Monday').'<br />';                  //输出上周一的时间戳
    echo strtotime('+1 week').'<br />';                      //输出下周此时此刻的时间戳
    echo strtotime('+1 day').'<br />';                       //输出明天此时此刻的时间戳
    echo strtotime('+1 week 2 days 4 hours 2 minutes 3 seconds').'<br />';
    echo strtotime('next Sunday').'<br />';                  //输出下周日此时此刻的时间戳
    echo strtotime('2017/09/23').'<br />';                   //输出 2017-9-23 的时间戳
    echo strtotime('2017-9-23').'<br />';                    //输出 2017-9-23 的时间戳
    echo strtotime('09/23/2017').'<br />';                   //输出 2017-9-23 的时间戳
    echo date('Y-m-d',strtotime('now')).'<br />';            //将现在的时间戳作为 date()函
                                                               数的参数
?>
```

5.5 自定义函数

虽然 PHP 提供了非常多的系统函数，大大地提高了开发程序的效率。但是这些系统函数还不能解决很多问题。对于新的环境、程序的个性化需求，系统函数就无能为力了，这时候需要用到自定义函数。

5.5.1 自定义函数的创建

创建自定义函数，可以方便程序开发，减少代码冗余。在程序需要修改时只需要重新修改自定义函数即可，无须到处修改代码。PHP 函数分为有返回值和无返回值两种，自定义函数也是这样。在 PHP 中使用 function 关键字来创建自定义函数。基本语法为：

```
function function_name($arg1[,$arg2[...[,argn]]]){
  //函数体
  [return $exspress]
}
```

上述语法格式中，function_name 是自定义函数名，不能和系统函数同名，符合 PHP 标识符命名规则。$arg1、$arg2、…、$argn 是函数的参数。函数可以不含参数（此时括号仍然不能少），也可以包含多个参数，包含多个参数时，参数与参数之间使用逗号隔开。大括号中间部分是函数体，也就是函数要执行的语句；return 语句是可选的，用来指定函数要返回的内容（表达式），该值可以是标准数据类型，还可以是数组。

说明：定义函数时的参数叫作形式参数，在调用函数时的参数叫作实际参数。

【示例 11】 自定义函数应用举例。

eg11.php 代码如下。

```
<?php
  header('Content-type:text/html;charset=utf-8');
  function login ($user, $password) {                           //$user 和 $password 是形式
参数
     if (($user == 'admin') && ($password == '123456' ))
     {
      echo "$user,You are welcome.";
```

```
        return true;
      }
      else
      {
        return false;
      }
  }
?>
```

5.5.2　自定义函数的调用

自定义函数要先定义再调用。调用语法为：

function_name($arg1[,$arg2[...[,argn]]])

【示例 12】 以调用示例 11 中的函数为例，看看自定义函数的调用方法。

步骤 1　eg12-1.php 代码如下。

```
<?php
  header('Content-type:text/html;charset=utf-8');
  function login($user,$password){                          //$user 和 $password 是形式
参数
      if (($user == 'admin') && ($password == '123456' ))
      {
        echo "$user,You are welcome.";
        return true;
      }
      else
      {
        return false;
      }
  }
  $name = 'admin';
  $pass = '12346';
  login($name,$pass) or die('非法用户');                    //$name 和 $pass 是实际参数
?>
```

程序运行结果如图 5-1 所示。

步骤 2　eg12-2.php 代码如下。

```
<?php
  header('Content-type:text/html;charset=utf-8');
  function login($user,$password){
      if (($user == 'admin') && ($password == '123456' ))
      {
        echo "$user,You are welcome.";
        return true;
      }
      else
      {
        return false;
```

```
        }
    }

    $name = 'admin';
    $pass = '123456';
    login($name,$pass) or die('非法用户');
?>
```

程序运行结果如图 5-2 所示。



图 5-1 eg12-1.php 的程序运行结果



图 5-2 eg12-2.php 的程序运行结果

说明：

- eg12-1.php 中，将两个实际参数 $name(= 'admin')和 $pass(= '12346')分别传递给自定义函数中的两个形式参数 $user 和 $password。即 $user= 'admin'和 $password='12346'，运行结果是函数体中的 if 语句不成立，login($name，$pass)返回 false，表达式"login($name，$pass) or die('非法用户')"的值由 die('非法用户')决定，因此执行 die()函数，显示"非法用户"信息。
- eg12-2.php 中，将两个实际参数 $name(= 'admin')和 $pass(= '123456')分别传递给自定义函数中的两个形式参数 $user 和 $password。即 $user= 'admin'和 $password='123456'，运行结果是函数体中的 if 语句成立，返回 true，根据短路运算规律，表达式"login($name，$pass) or die('非法用户')"不执行后面的 die()函数，直接显示"admin，You are welcome."信息。

5.5.3 参数传递

在调用函数时，用户信息是通过函数参数来传递给函数本身的。函数可以不带参数，也可以带一个或多个参数。PHP 支持的参数传递方式有：按值传递参数、按引用传递参数、默认参数值和可变参数。

1. 按值传递参数

按值传递参数是 PHP 函数中默认的也是最常见的参数传递方式。这种方式仅仅是把函数外部实际参数的值(副本)传递给函数的形式参数。函数处理后(无论函数里面做了什么处理)，都不会影响到实际参数本身。这种参数传递是单向的。实际参数可以是表达式。

【示例 13】 按值传递参数应用举例。

eg13.php 代码如下。

```
<?php
    function add($arg1,$arg2){
        $sum = $arg1 + $arg2;
        $arg1 = $arg1 + 10;
        return $sum;
```

```
    }
    $one = 12;
    $two = 23.4;
    echo add($one,$two).'<br />';                     //输出 35.4
    echo $one.'<br />';                               //输出 12,而不是 22
    //下面的实际参数是表达式
    echo add(2+31,2*11);                              //输出 55
?>
```

说明：

- add($ one, $ two)在调用 add()函数后，$ one 没有随着 $ arg1 值的变化而变化，实际上 $ one 和 $ two 都是单向传递的，add()里面对 $ arg1 和 $ arg2 的改变不会影响 $ one 和 $ two。
- add(2＋31，2 * 11)在调用时也是把 33 和 22 的值分别传递给 $ arg1 和 $ arg2，此处两个实际参数是表达式。

2. 按引用传递参数

按引用传递就是按照地址来传递：实际参数只能是变量(因为表达式没有地址)，实际参数把地址传递给形式参数(形式参数和实际参数实际上是不同名称的两个相同变量)。换言之，形式参数的改变是会影响到实际参数的，因为形式参数和实际参数为同一个变量，它们具有相同的地址单元。

为了和区别值传递，在使用引用传递时要在形式参数前面冠以“&”符号。

【示例 14】 按引用传递参数应用举例。

```
<?php
    function add(&$arg1,$arg2){                       //&$arg1 引用传递
      $sum = $arg1 + $arg2;
      $arg1 = $arg1 + 10;
      return $sum;
    }
    $one = 12;
    $two = 23.4;
    echo add($one,$two).'<br />';                     //输出 35.4
    echo $one.'<br />';                               //输出 22,而不是 12
    //echo add(2+31,2*11);                            //出错
?>
```

说明：

- & $ arg1 采用引用传递，在执行语句“add($ one, $ two)”时把实际参数 $ one 的地址传递给 $ arg1，则 $ arg1 和 $ one 就是一个变量，$ arg1 的修改影响到了 $ one，所以 $ one 的值变成了 22。
- 形如“add(2＋31，2 * 11)”的调用将出错，因为对应于 $ arg1 的实际参数只能是变量，而不能是形如 “2＋31”的表达式。

有时候会利用引用传递的双向性来“做文章”，使函数的功能更强大。

【示例 15】 定义一个自定义函数来求三个数的最大值和最小值。

分析：因为通过函数名只能返回一个结果，不妨让函数名返回最大值，再通过引用传递

参数方式来得到最小值，一举两得。根据分析，eg15. php 代码如下。

```
<?php
  header('Content-type:text/html;charset=utf-8');
  function zuida($arg1,$arg2,$arg3,&$min){
    $max=$arg1;
    if ($arg2>$max) $max=$arg2;
    if ($arg3>$max) $max=$arg3;
    $min=$arg1;
    if ($arg2<$min) $min=$arg2;
    if ($arg3<$min) $min=$arg3;
    return $max;
  }
  $a=31;
  $b=12;
  $c=3;
  echo '最大数：'.zuida($a,$b,$c,$d).'<br />';
  echo "最小数：$d";
?>
```

说明：

- 函数 zuida()是通过函数名来返回最大值的。
- 函数 zuida()中的第四个形式参数的定义为引用传递参数方式，函数中将最小值赋值给该参数，该参数会反向传递给实际参数 $d，因此 $d 就是最小值。
- 在实际参数中，第一、二、三个参数（即 $a、$b、$c）可以是表达式，而第四个参数（即 $d）只能是变量。

3. 默认参数值

自定义函数中可以使用预先定义好的默认参数值。在调用时，若未指定参数值，则使用默认参数值；若明确指定了参数值，则使用指定的参数值。

【示例 16】 默认参数值应用举例。

eg16. php 代码如下。

```
<?php
  function showInfo($name,$sex='male',$age=18){
    echo "Your name is $name,your sex is $sex,your age is $age.<br />";
  }
  showInfo('Liubei');                                  //两个默认参数值
  showInfo('Diaochan','female');                       //一个默认参数值
  showInfo('Xiaoqiao','female',19);                    //不使用默认参数值
?>
```

下面的写法是不提倡的：

```
<?php
  /*  $sex 不能放置在$name 之前，因为有默认参数值的形式参数只能放置在后，否则会产生一个
警告信息 */
  function showInfo($sex='male',$name,$age=18){
    echo "Your name is $name,your sex is $sex,your age is $age.<br />";
  }
```

```
   //其他代码
?>
```

eg16.php 的程序运行结果如图 5-3 所示。

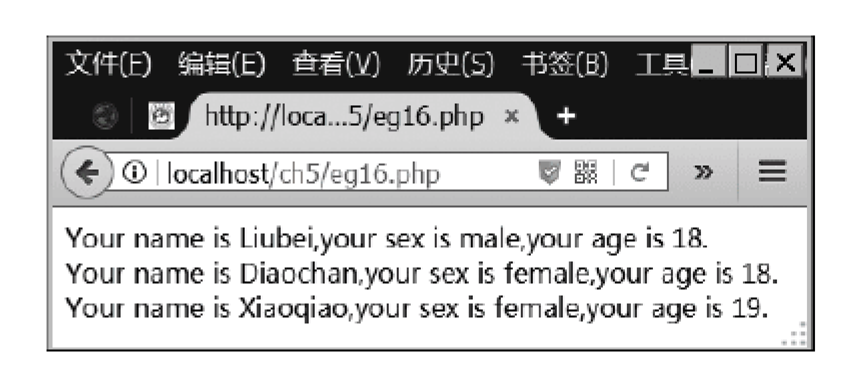


图 5-3 示例 16 的程序运行结果

说明：

- 函数的默认参数可以是多个，此时应该把这些默认参数放置在函数的最后。
- 在调用函数时，如果实际参数的个数小于形式参数的个数，PHP 会按顺序来把实际参数传递给形式参数，形式参数中的多余部分则使用默认参数值来代替。

4. 可变参数

所谓可变参数，就是函数的参数个数是可以变化的。函数能够根据传入的不同参数进行不同的处理。在可变参数的函数中经常要用到 3 个系统函数。

（1）func_get_arg()函数

func_get_arg()函数的语法格式为：

```
mixed func_get_arg ( int $arg_num )
```

从用户自定义函数的参数列表中获取某个指定的参数。该函数可以配合 func_get_args()和 func_num_args()一起使用，从而使用户自定义函数可以接受自定义个数的参数列表。arg_num 是参数的偏移量。函数的参数是从 0 开始计数的。

返回值：返回指定的参数，有错误则返回 false。

（2）func_get_args()函数

func_get_args()函数的语法格式为：

```
array func_get_args ( void )
```

获取函数参数列表的数组。该函数可以配合 func_get_arg() 和 func_num_args() 一起使用，从而使用户自定义函数可以接受自定义个数的参数列表。

返回值：返回一个数组，其中每个元素都是目前用户自定义函数的参数列表的相应元素的副本。

（3）func_num_args()函数

func_num_args()函数的语法格式为：

```
int func_num_args ( void )
```

获得参数的个数，常与函数 func_get_arg()和 func_get_args()一起使用，用于获取函数的参数个数。

返回值：自定义函数的参数个数。

【示例 17】 可变参数的函数应用举例。

eg17. php 代码如下。

```
<?php
  function foo()
  {
    $numargs = func_num_args();          //参数个数
    echo "Number of arguments:$numargs.<br />";
    if ($numargs >=2){
      echo "Second argument is: ". func_get_arg(1)."<br />";
      //func_get_arg(1)表示第 2 个参数
    }
    $arg_list = func_get_args();         //获得参数列表(数组类型)
    for($i =0;$i<$numargs; $i++){
      echo "Argument $i is: ".$arg_list[$i]."<br />";
    }
  }
  foo('Liubei','male',23,'01');          //4 个参数
?>
```

程序运行结果如图 5-4 所示。

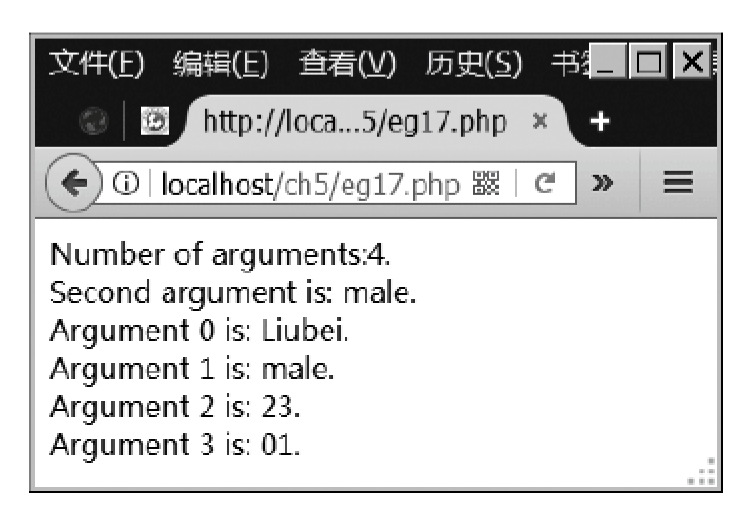


图 5-4 示例 17 的程序运行结果

【示例 18】 编写一个函数,用于求几个数的和,要求使用可变参数。

eg18. php 代码如下。

```
<?php
  function add()
  {
    $sum =0;
    $numargs = func_num_args ();
    $arg_list = func_get_args();
    for($i =0;$i<$numargs; $i++){
      $sum +=$arg_list[$i] ;
    }
    return $sum;
  }
  echo add(1).'<br />';                  //输出 1
  echo add(1,2).'<br />';                //输出 3
  echo add(1,2,3).'<br />';              //输出 6
?>
```

5.5.4　函数的返回值

一般情况下，如果只靠函数做某些事情还是远远不够的。有时候我们希望函数能够进行数据方面的处理，并把结果返回给调用程序，因为涉及变量的作用域，有时候往往不能通过函数的局部变量来将数据返回给调用语句，这时候可以使用 return 语句来将值返回给调用语句。return 既可以返回单个的值，也可以返回多个值，比如数组，甚至还可以返回对象。

1. 返回单个值

通过 return 关键字向函数的调用者返回值，将程序控制权返回到调用者，因此该返回值回到了调用者的作用域。使用 return 返回单个值最常见，非常简单。

【示例 19】　编写一个函数，用于求 sum＝1＋2＋3＋4＋…＋n。

eg19. php 代码如下。

```
<?php
  function sum($n){
     $s =0;
     for($i=1;$i<=$n;$i++)
       $s+=$i;
     return $s;
  }
  echo sum(100);                        //输出为 5050
?>
```

2. 返回数组

要想让自定义函数返回多个值，可以通过返回数组来实现，然后再从数组中获得各个数组元素即可。

【示例 20】　编写一个自定义函数，用于求用户的个人信息，假设用户的信息包括 ID、姓名、性别和年龄。

eg20. php 代码如下。

```
<?php
  $a =array( array('id'=>'01','name'=>'Liubei' ,'sex'=>'male' ,'age'=>23),
             array('id'=>'02','name'=>'Diaochan','sex'=>'female','age'=>21),
             array('id'=>'03','name'=>'Guanyu','sex'=>'male' ,'age'=>22),
             array('id'=>'04','name'=>'Zhangei','sex'=>'male ,'age'=>20),
             array('id'=>'05','name'=>'Xiaoqiao','sex'=>'female','age'=>19)
         );
  //$a 是二维数组，每一个元素又是一个一维数组，表示某个人的个人信息
  function getUserInfo($id){
    global $a;
    $user =array();
    foreach($a as $value){      //遍历数组的每一个元素
      if ( $value['id'] ==$id){//若某数组元素的 id 等于$id,则把该元素赋值到数组$user 中
          $user['id'] =$value['id'];
          $user['name'] =$value['name'];
          $user['sex'] =$value['sex'];
          $user['age'] =$value['age'];
      }
```

```
    }
    return $user;
  }
  $user = getUserInfo("03");    //获得 id='03'的个人信息
  echo $user['id'].'---'.$user['name'].'---'.$user['sex'].'---'.$user['age'];
  //输出为 Guanyu 的个人信息
?>
```

说明：

- getUserInfo($id)函数用于从二维数组中获得某个元素，该元素是一个一维数组，表示 id=$id 的人的信息。
- 因为 $a 这个二维数组定义在函数 getUserInfo($id)外面，getUserInfo($id)函数要想使用该数组，需声明该数组为全局变量。
- getUserInfo($id)函数返回的是数组，数组的值分别是'id'、'name'、'age'和'sex'。

3. 返回对象

PHP 自定义函数还可以返回一个对象。调用函数并返回对象之后，可以使用该对象的属性和方法来实现某些功能。

【示例 21】 本例通过自定义函数 getInfo($name, $age)来获得 Student 类的一个对象，并将函数的返回结果赋值给变量 $stu。$stu 变量实际上就是对象，可以调用 $stu 对象的 show()方法来显示学生的信息。

eg21.php 代码如下。

```
<?php
  class Student{
     private $name,$age;
     function __construct($name,$age){
        $this->name = $name;
        $this->age = $age;
     }
     function show(){
        echo "My name is $this->name,I'm $this->age.";
     }
  }
  function getInfo($name,$age){
    $st=new Student($name,$age);
    return $st;
  }
  $stu=getInfo("Liubei",23);
  $stu->show();
?>
```

5.6 函数应用

前面几节介绍了自定义函数的创建和调用以及参数的传递等知识。本节继续讲解自定义函数的其他应用，包括变量函数、嵌套函数和递归函数。

5.6.1　变量函数

程序员在编写程序时，如果要处理的问题比较复杂，需要创建多个函数。为了调用这些函数，需要记住这些函数的函数名以及函数的作用。换言之，就是程序员看见函数名称，就能知道该函数的功能，然后调用这些函数。

PHP 支持一种所谓的变量函数。变量函数是指这个函数的名称是变量，需要等到执行语句计算出该变量值时才知道要调用哪个函数。变量函数和变量一样需要在函数前面加 $ 符号。这意味着如果一个变量名后有圆括号，PHP 将寻找与变量的值同名的函数，并且将尝试执行它。除了别的事情以外，这个可以用于实现回调函数及函数表等。变量函数不能用于语言结构，例如 echo()、print()、unset()、isset()、empty()、include()、require() 以及类似的语句。需要使用自己的外壳函数来将这些结构用作变量函数。

【示例 22】　变量函数应用举例。

eg22.php 代码如下。

```
<?php
  function foo() {
    echo "In foo() function.<br />";
  }
  function bar($arg) {
    echo $arg;
  }
  $func = 'foo';
  $func();                              //调用 foo()函数
  $func = 'bar';
  $func('In bar() function.<br />');    //调用 bar()函数
?>
```

【示例 23】　还可以利用变量函数的特性来调用一个对象的方法。

eg23.php 代码如下。

```
<?php
  class Foo
  {
    function Variable()
    {
      $name = 'Bar';
      $this->$name();                   //调用 Bar()方法
    }
    function Bar()
    {
      echo "This is Bar";
    }
  }
  $foo = new Foo();
  $funcname = "Variable";
  $foo->$funcname();                    //调用 Foo 类中的 Variable 方法
?>
```

5.6.2 嵌套函数

所谓嵌套函数,就是在函数中再定义其他函数,也就是可以在函数里面调用其他函数。比如求组合数 C(m,n),我们知道 C(m,n)=m! /((m-n)! * n!)。因为求组合数需要用到求阶乘,因此可以定义一个自定义函数来求组合数,然后再在组合数函数中定义求阶乘函数。

【示例 24】 定义自定义函数求组合数,在组合数函数中再定义求阶乘函数。

步骤 1 eg24-1. php 代码如下。

```
<?php
  function comb($m,$n){
    function fac($k){
      $mul =1;
      for($i=1;$i<=$k;$i++)
        $mul =$mul * $i;
      return $mul;
    }
    $result =fac($m)/(fac($m-$n) * fac($n));
    return $result;
  }
  echo comb(5,3).'<br />';
  echo fac(3).'<br />';        //先调用 comb1(5,3),然后才能调用 fac(3)
  echo comb(5,2).'<br />';     //出错,系统认为是重复定义了 fac()函数
?>
```

说明:

- fac()函数定义在 comb()里面,因此必须先调用 comb()函数之后才能调用 fac()函数,否则会被认为 fac()函数没有定义。
- 在第 2 次调用 comb()函数时系统显示重复定义了 fac()函数。原因是第 1 次调用 comb()函数是就把 fac()函数定义了,所以第 2 次调用 comb()函数时就出现重复定义 fac()函数的现象。
- 在 comb()函数中先定义 fac()函数,再调用 fac()函数,顺序不可以颠倒。

步骤 2 在 eg24-1. php 中,第 2 次调用 comb()函数时会显示 fac()已经定义的错误信息。为了解决 comb()函数多次调用不出错的问题,可以先判断一下 fac()函数是否已经定义,如果没有定义再来定义。eg24-2. php 代码如下。

```
<?php
  function comb($m,$n){
    if (!function_exists('fac')){
      function fac($k){
        $mul =1;
        for($i=1;$i<=$k;$i++)
          $mul =$mul * $i;
        return $mul;
      }
```

```
    }
    $result = fac($m)/(fac($m-$n) * fac($n));
    return $result;
  }
  echo comb(5,3).'<br />';
  echo fac(3).'<br />';
  echo comb(5,2).'<br />';
?>
```

5.6.3　递归函数

PHP 支持函数的递归调用。所谓函数的递归调用，简言之就是递归函数，即函数直接或间接地调用它自己。在科学技术领域有些问题使用递归的方法往往可以迎刃而解，反之解决起来比较困难。

递归的思路是把一个复杂的问题分解成简单的且相同的问题，再加有限操作来加以实现。换言之就是把一个规模大一点的问题分解成为规模小一点的问题，再加有限的操作。

在 PHP 中递归的层数是有限的。PHP 5 理论上可以支持 6000 个递归调用，但是往往又与计算机的软硬件密切相关。

【示例 25】 用自定义函数求阶乘，要求使用递归的方法。

eg25.php 代码如下。

```
<?php
  function fac($n){
    if ($n==0)
       return 1;
    else
       return fac($n-1) * $n;
  }
  echo fac(10);
?>
```

还可以写成

```
<?php
  function fac($n){
    if ($n==0)
       $result=1;
    else
       $result=fac($n-1) * $n;
    return $result;
  }
  echo fac(10);
?>
```

说明：

- 要想求 n 的阶乘，只需要求 $n-1$ 的阶乘即可，然后再把 $n-1$ 的阶乘乘以 n；要想求 $n-1$ 的阶乘，只需要求 $n-2$ 的阶乘即可，然后再把 $n-2$ 的阶乘乘以 $n-1$ 即可。以此类推，于是求 n 的阶乘就变成了求 0 的阶乘，而 0 的阶乘可以直接给出为 1。

• 程序中“return fac($ n－1) * $ n;”和“ $ result = fac($ n－1) * $ n;”语句就是自己调用自己，即递归调用。

5.7 综合案例——汉诺塔问题

汉诺塔问题是一个经典的问题。汉诺塔(Hanoi Tower)又称河内塔，源于印度一个古老传说。大梵天创造世界时做了三根金刚石柱子，在一根柱子上面从下往上按照大小顺序摞着 64 片黄金圆盘。大梵天命令婆罗门把圆盘从下面开始按大小顺序重新摆放在另一根柱子上。并且规定，任何时候，在小圆盘上都不能放大圆盘，且在三根柱子之间一次只能移动一个圆盘。问应该如何操作？汉诺塔问题如图 5-5 所示。

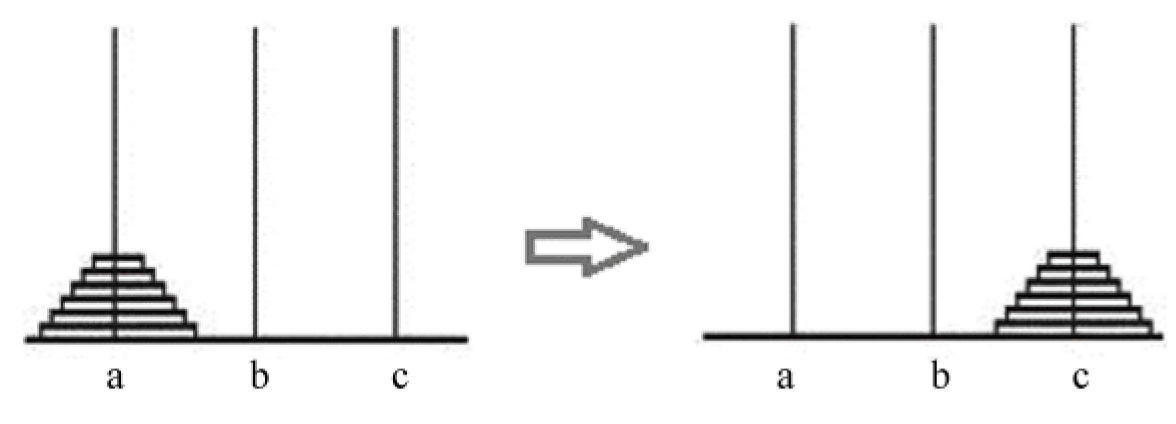


图 5-5　汉诺塔问题

步骤 1　首先要分析问题：这个问题不宜使用一般的办法来实现，凭感觉应该使用递归的知识来实现，一股脑地考虑每一步如何移动很困难，我们可以换个思路。先假设除最下面的盘子之外，我们已经成功地将上面的 63 个盘子移到了 b 柱，此时只要将最下面的盘子由 a 移动到 c 即可，如图 5-6 所示。

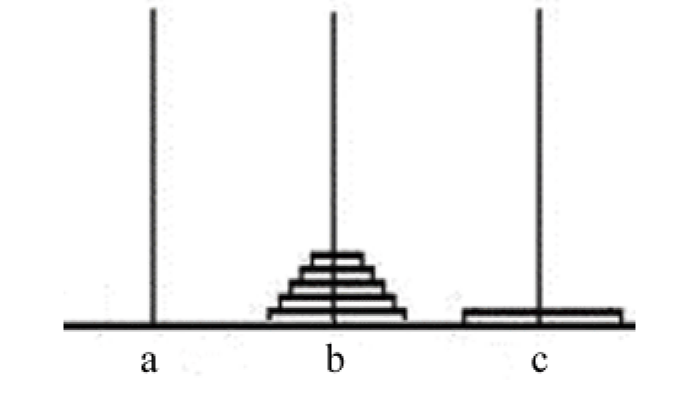


图 5-6　简化为规模小的同类问题

当最大的盘子由 a 移动到 c 后，b 上是余下的 63 个盘子，a 为空。因此现在的目标就变成了将这 63 个盘子由 b 移动到 c。这个问题和原来的问题完全一样，只是由 a 柱换为 b 柱，规模由 64 变为 63。因此可以采用相同的方法，先将上面的 62 个盘子由 b 移动到 a，再将最下面的盘子移动到 c……对照下面的过程，试着是否能找到规律：

① 将 b 柱子作为辅助，把 a 上的 63 个圆盘移动到 b 上。

② 将 a 上最后一个圆盘移动到 c 上。

③ 将 a 作为辅助，把 b 上的 62 个圆盘移动到 a 上。

④ 将 b 上的最后一个圆盘移动到 c 上。

……

也许你已经发现规律了，即每次都是先将其他圆盘移动到辅助柱子上，并将最底下的圆盘移到 c 柱上，然后再把原先的柱子作为辅助柱子，并重复此过程。

这个过程称为递归，即定义一组基本操作，这组操作将规模小一点(或大一点)的操作当作一个整体——无须关心它的细节，只当它已经完成了——然后执行剩下的操作。而在更小或更大的规模中也依此操作，直到规模达到预定值(规模小到预定值可以简单完成)。

根据分析，若定义函数 hanno($a, $b, $c, $n)为汉诺塔函数，其中 $a 为原始石柱，$b 是过渡石柱，$c 是目标石柱，$n 是圆盘数量。则 hanno($a, $b, $c, $n)可以分为三步来完成。

第 1 步：将最上面 $n−1 个圆盘从 $a 移动到 $b，可以经过 $c 过渡，即 hanno($a, $c, $b, $n−1)。

第 2 步：将 $a 上的最大的一个圆盘移动到 $c，即 hanno($a, $b, $c, 1)。

第 3 步：将 $b 上全部 $n−1 个圆盘移动到 $c，可以经过 $a 过渡，即 hanno($b, $a, $c, $n−1)。

步骤 2　根据分析，汉诺塔问题 eg26.php 代码如下。

```
<?php
  $count = 0;     //定义计数用的全局变量，保存移动次数的信息
  function hannuo($a,$b,$c,$n){
    global $count;
    if ( $n>1 ){
      hannuo($a,$c,$b,$n-1);
      hannuo($a,$b,$c,1);
      hannuo($b,$a,$c,$n-1);
    } else {
      $count++;
      echo "$count. Move $a to $c.<br />";
    }
  }
  hannuo('a','b','c',3);     //将 3 个圆盘从 a 柱移动到 c 柱
?>
```

步骤 3　程序运行结果如图 5-7 所示。



图 5-7　汉诺塔的程序运行结果

5.8　习　　题

一、填空题

1. 在 PHP 中函数分为系统函数和________。
2. 数学函数________对浮点数实现四舍五入。
3. 在用户自定义函数时需要使用的关键字是________。

4. 用户自定义函数可以使用________关键字向调用者返回值。

5. 使用________函数可以获得此时此刻的时间戳。

6. 使用________函数可以获取今天是今年的第几天，今天是星期几，此时此刻的时、分、秒。

7. 使用________函数可以保留小数点后面 n 位。

8. 使用________函数可以求出多个数的最大值。

9. 使用________函数可以得到随机数。

10. 使用________函数可以格式化日期时间。

二、选择题

1. 下列函数中，________函数不用于判断变量是否为整数。

A. is_real()　　B. is_int()　　C. is_long()　　D. is _integer()

2. getdate()函数以数组的形式返回日期时间，使用________可以获得今天是星期几？

A. mday　　B. yday　　C. wday　　D. weekday

3. getdate()函数以数组的形式返回日期时间，使用________可以获得今天是一年的第几天？

A. mday　　B. yday　　C. wday　　D. weekday

4. 判断变量是否存在，使用的函数是________。

A. isset()　　B. unset()　　C. var_dump()　　D. is_array()

5. 关于函数的返回值，说法不正确的是________。

A. 函数只能返回一个变量　　B. 函数可以返回数组

C. 函数可以什么都不返回　　D. 函数可以返回对象

6. 关于函数的参数传递，下面说法正确的是________。

A. 函数的参数传递按照名称来传递　　B. 函数参数传递按照位置先后顺序传递

C. 函数的实参和形参个数可以不一样多　　D. 函数的参数不可以设置默认值

7. 关于函数，下列说法错误的是________。

A. 可以在函数中调用函数本身　　B. 可以在函数中定义函数

C. 函数的实参可以是另一个函数　　D. 函数名可以和标准函数同名

三、程序设计题

1. 定义一个函数 zuida()，该函数既可以用于求三个数的最大值，又可以用于求三个数的最小值。请使用返回数组的方法完成。

2. 定义一个函数 zuida()，该函数既可以用于求三个数的最大值，又可以用于求三个数的最小值。请使用引用参数传递的方法完成。

3. 定义函数，求一个数的因子和。

4. 定义函数，求 7 天后是一年的第几天。

5. 定义递归函数，求 $1+2+3+4+\cdots+n$。

6. 定义递归函数，解决汉诺塔问题。汉诺塔是根据一个传说形成的一个问题：有三根杆子 A、B、C。A 杆上有 N 个($N>1$)穿孔圆盘，盘的尺寸由下到上依次变小。要求按规则将所有圆盘移至 C 杆。规则是：可将圆盘临时置于 B 杆，也可将从 A 杆移出的圆盘重新移回 A 杆。问：如何移？最少要移动多少次？

第 6 章　数　　组

知识点：

- 数组的概念与分类
- 创建、访问和追加数组
- 修改和删除数组
- 遍历数组
- 对数组进行排序
- 数组的联合与合并
- 数组的拆分与替换
- 随机获取数组元素
- 数组的其他操作

本章导读：

在程序开发的过程中，有时候需要创建很多相似的变量，对于这些相似变量可以把数据作为元素存储在数组中。PHP 数组功能非常强大，系统提供了非常多的函数用来处理数组，使数组简单、灵活，使用方便。

程序员可以把文本文件、XML 文件存储到数组中，还可以把数据库导出的记录存储在数组中，在内存中用数组来代替复杂的数据库查询，可以减少数据库的输入/输出，从而提高了程序的性能。

6.1　数组概述

PHP 中的数组操作是非常重要的内容，PHP 程序员经常需要和数组打交道，因此了解与掌握数组是相当重要的。本节将介绍数组的一些基本的概念。

6.1.1　数组的概念

数组，顾名思义就是数据的组合。它是一系列同名的变量，为了批量处理而构建的一种数据类型。数组中的每一个元素都有值(value)和键(key)两个属性。

PHP 中的数组实际上是一个有序映射。映射是一种把 values 关联到 keys 的类型。此类型在很多方面做了优化，因此可以把它当成真正的数组，或列表(向量)、散列表(是映射的一种实现)、字典、集合、栈、队列以及更多可能性。由于数组元素的值也可以是另一个数组，树形结构和多维数组也是允许的。

例如，数组：

```
array('name'=>'Liubei','sex'=>'male','age'=>23)
```

键：

```
'name','sex','age'
```

值：

```
'liubei','male',21
```

6.1.2 数组的分类

1. 根据索引分类

在 PHP 中数组的键可以是数字，也可以是字符串，而不像其他语言中数组的键必须是数字。根据数组的键可以把数组分成数字索引数组和关联数组。

（1）数字索引数组

以数字作为数组的键。PHP 中数组的键默认是从 0 开始的，然后自动加 1，不需要特别说明。程序员还可以指定数组的键。

例如：

```
<?php
  header("content-type:text/html;charset=utf-8");
  $a1 =array("Beijing","Shanghai","Guangzhou","Shenzhen");
  echo $a1[0].'--'.$a1[1].'--'.$a1[2].'--'.$a1[3];
?>
```

将会输出：

```
Beijing--Shanghai--Guangzhou--Shenzhen
```

【示例 1】 数字索引数组举例。

eg1.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $a1[] = 'Beijing';            //键值 0
  $a1[] = 'Shanghai';           //键值 1
  $a1[] = 'Guangzhou';          //键值 2
  $a1[] = 'Shenzhen';           //键值 3
  $a1[2]='广州';                //修改 a1[2]为'广州'
  echo $a1[0].'--'.$a1[1].'--'.$a1[2].'--'.$a1[3].'<br />';
  $a2[3] = 'Liubei';            //键值 3
  $a2[] = 'Guanyu';             //键值 4
  $a2[] = 'Zhangfei';           //键值 5
  $a2[] = 'Zhaoyun';            //键值 6
  $a2[4]='Guanyunchang';        //键值 4,修改 Guanyu 为 Guanyunchang
  echo $a2[3].'--'.$a2[4].'--'.$a2[5].'--'.$a2[6].'<br />';
?>
```

说明：

- PHP 数组键值默认从 0 开始，然后自动加 1；
- PHP 开始键值也可以指定，后面会自动加 1。

（2）关联数组

以字符串或者字符串和数字混合作为键名的数组称为关联数组（Associative Array）。关联数组的键名可以是数字或者字符串的混合形式，而不像数字索引数组只能为数字。

例如：

```
<?php
  $a['name'] = 'Liubei';
  $a['sex'] = 'male';
  $a['age'] =23;
  $a['phone']='13088888888';
  $a[1] =87;
  $b =array('Yuwen'=>98,'Shuxue'=>88,'Yinggyu'=>99,'Wuli'=>87);
?>
```

说明：上面所定义的数组 $a 是关联数组，数组的键以字符串命名，这种数组可读性比较强，在现实中非常常见。

2. 根据维度分类

数组根据维度分类，可以分为一维数组、二维数组和多维数组。超过二维数组就是多维数组。

（1）一维数组

一维数组是最普通的数组，它只保存一列数据。数组的每一个元素不再是数组，即数组的每一个元素只需要使用一个索引即可访问。

例如，表示一个学生的姓名、性别、年龄、电话和语文成绩，可以表示为：

```
$a['name'] = 'Liubei';
$a['sex'] = 'male';
$a['age'] =23;
$a['phone']='13088888888';
$a[1] =87;
```

（2）二维数组

二维数组就是要使用两个索引来表示一个最基本的元素，或者说数组的元素又是一个一维数组。例如，a[m][n]表示二维数组，可以理解为 a 数组有 m 个元素，每个元素又是一个有 n 个元素的一维数组。

例如，表示某班学生的信息的数组可以表示为：

```
$a[0]['name'] ='Liubei';
$a[0]['sex'] ='male';
$a[0]['age'] =23;
$a[0]['phone']='13088888888';
$a[0][1] =87;
$a[1]['name'] =Guanyu;
$a[1]['sex'] ='male';
```

```
$a[1]['age'] =22;
$a[1]['phone']='13011111111';
$a[1][1] =81;
$a[2]['name'] ='Diaochan';
$a[2]['sex'] ='female';
$a[2]['age'] =20;
$a[2]['phone']='13022222222';
$a[2][1] =78;
```

说明：

- 上面是一个二维数组，每一个基本信息需要使用两个索引来表示，例如表示 Guanyu 的年龄需要使用 $a[1]['age']。
- 上面数组表示了多个人的信息，每个人的信息又是一个一维数组，包括姓名、性别、年龄、电话和语文成绩。

(3) 多维数组

在 PHP 中，可以创建多维数组，例如三维数组、四位数组等。但是真正在应用中很少使用多维数组，因为数组随着维数的增加使用起来会越来越复杂。

读者可以自己尝试写出一个多维数组。

6.2 数组的基本操作

在 PHP 中提供了很多操作方法或者函数用于对数组进行操作，这使对数组的创建、访问、修改和删除等操作变得非常容易。

6.2.1 数组的创建

PHP 中数组是比较松散的数据类型，与 C 语言或者 Java 语言不相同，它无须先声明，而是可以直接创建。

在 PHP 中要创建数组可以直接给数组元素赋值，还可以使用 array()函数创建数组。此外还可以使用 range()函数创建数组。

1. 直接赋值创建数组

这种方式最简单，只需要指明键和值即可。格式如下。

```
$array_name[key]=vaule;
```

说明：键名 key 也可以省略。若省略则默认从 0 开始，自动加 1。

例如：

```
$a1[0] =89;
$a1[1] =98;
```

还可以为：

```
$a1[] =89;
$a1[] =98;
```

【示例 2】 直接赋值创建数组。

eg2. php 代码如下。

```
<?php
  $a1[] =89;                //key 为 0
  $a1[] =98;                //key 为 1
  $a2[3] ='liubei';         //key 为 3
  $a2[] ='Guanyu';          //key 为 4
  $a3[] =88;                //key 为 0
  $a3[4] =66;               //key 为 4
  $a3[] =55;                //key 为 5
  $a4['name'] ='Liubei';    //key 为 name
  $a4['sex'] ='male';       //key 为 sex
  $a4['age'] =23;           //key 为 age
  var_dump($a1);            //array(2) { [0]=>int(89) [1]=>int(98) }
  var_dump($a2);            //array(2) { [3]=>string(6) "liubei" [4]=>string(6) "Guanyu" }
  var_dump($a3);            //array(3) { [0]=>int(88) [4]=>int(66) [5]=>int(55) }
  var_dump($a4);
  //array(3) {["name"]=>string(6)"Liubei"["sex"]=>string(4)"male"["age"]=>int(23)}
?>
```

说明：

- 数组的 key 默认从 0 开始，自动加 1，也可以指定初始 key 值；
- 在容易出错的情况下，可以明文指定数组的 key。

2. 使用 array()函数创建数组

使用 array()函数可以创建数组。array()函数的语法格式为：

```
array array ([mixed $ ...] )
```

参数格式为 value 或者 key=>value。

返回值：数组类型。

【示例 3】 使用 array()函数创建数组。

eg3. php 代码如下。

```
<?php
  $sanguo=array('Liubei','Guanyu','Zhangfei');
  $shuihu=array( array('id'=>'01','name'=>'Songjiang'),
                 array('id'=>'02'),);
  $xiyou=array('Sunwukong','master'=>'Tangseng','Zhubajie','Shaseng');
  $a=array(67,89,98);
  print_r($sanguo);         //array([0]=>Liubei[1]=>Guanyu[2]=>Zhangfei)
  print_r($shuihu);
  //array([0]=>array([id]=>01[name]=>Songjiang)[1]=>array([id]=>02))
  print_r($xiyou);
  //array([0]=>Sunwukong [master]=>Tangseng [1]=>Zhubajie [2]=>Shaseng)
  print_r($a);              //array([0]=>67 [1]=>89 [2]=>98)
?>
```

说明：

- 使用 array()函数可以创建一维数组、二维数组和多维数组。
- array()函数的参数可以是 value 或者 key=>value 两种形式。若不指定 key,则默认从 0 开始并自动加 1。
- print_r()函数用于输出数组。

3. 使用 range()函数创建数组

在 PHP 中可以使用 range()函数来创建数组。语法格式为:

```
array range ( mixed $start, mixed $limit [, number $step =1 ] )
```

说明:建立一个包含指定范围单元的数组。返回值是数组类型,返回的数组为从 $start 到 $limit 的单元,包括它们本身。参数 $start 表示起始数,即数组的第 1 个值;参数 $limit 表示序列结束于 $limit;$step 是可选参数,如果给出了 $step 的值,它将被作为单元之间的步进值。$step 应该为正值,如果未指定,$step 则默认为 1。

【示例 4】 使用 range()函数创建数组。

eg4.php 代码如下。

```
<?php
  $a1 =range('A','Z',5);
  $a2 =range(5,10);
  $a3 =range(1,20,4);
  print_r($a1);
  //Array ([0]=>A [1]=>F [2]=>K [3]=>P [4]=>U [5]=>Z)
  print_r($a2);
  //Array ([0]=>5 [1]=>6 [2]=>7 [3]=>8 [4]=>9 [5]=>10)
  print_r($a3);
  //Array ([0]=>1 [1]=>5 [2]=>9 [3]=>13 [4]=>17)
?>
```

说明:

- range()函数创建的数组不仅可以是数值类型,还可以是字符类型。
- 创建的数组包含初始值、结束值(与步长有关系),省略 $step 参数时,则默认步长为 1。

6.2.2 数组元素的追加

在 PHP 中除了可以直接给数组元素赋值来进行数组元素添加之外,还可以使用 array_unshift()函数和 array_push()函数进行数组元素的添加。

1. 使用 array_unshift()函数进行数组元素的添加

在 PHP 中可以使用 array_unshift()函数进行数组元素的添加。array_unshift()函数用于在数组开头插入一个或多个单元。

array_unshift()函数的语法格式为:

```
int array_unshift (array &$array,mixed $var [,mixed $ ...])
```

说明:array_unshift()将传入的单元插入 array 数组的开头。注意单元是作为整体被插入的,因此,传入单元将保持同样的顺序。所有的数值键名将修改为从零开始重新计数,

所有的字符键名保持不变。返回值是新数组的元素个数。

【示例 5】 使用 array_unshift()函数进行数组元素的添加。

eg5.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $city =array('河南'=>'郑州','南京','湖北'=>'武汉','长沙');
  //array(4){["河南"]=>string(6)"郑州" [0]=>string(6)"南京" ["湖北"]=>string(6)"武汉"
  [1]=>string(6)"长沙"}
  var_dump($city);
  echo '<br />';
  $n =array_unshift($city,'广州','成都');
  var_dump($city,$n);
  /*
     array(6){[0]=>string(6)"广州"  [1]=>string(6)"成都"  ["河南"]=>string(6)"郑州"[2]
     =>string(6)"南京"  ["湖北"]=>string(6)"武汉"  [3]=>string(6) "长沙" }   int(6)
  /*
?>
```

说明：

- array_unshift()函数将'广州'和'成都'加到数组 $city 最前面，不改变新添加数组的顺序，并且改变数组 $city 的数值键，重新从 0 开始。数字键自动加 1，字符键保持不变。
- 增加数组元素之后，$city 数组元素变成 7 个，array_unshift()函数的返回值就是新数组的数组元素个数。

2. 使用 array_push()函数进行数组元素的添加

在 PHP 中可以使用 array_push()函数进行数组元素的添加。array_push()函数用于将一个或多个单元压入数组的末尾(入栈)。array_push()函数的语法格式为：

```
int array_push ( array &$array, mixed $var [, mixed $ ... ] )
```

说明：array_push()将 array 当成一个栈，并将传入的变量压入 array 的末尾。array 的长度将根据入栈变量的数目增加。返回值是新数组元素的个数。

【示例 6】 使用 array_push()函数进行数组元素的添加。

eg6.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $city =array('河南'=>'郑州','南京');
  var_dump($city);
  //array(2) {["河南"]=>string(6)"郑州" [0]=>string(6)"南京" }
  echo '<br />';
  $n =array_push($city,'广州','成都');
  var_dump($city,$n);
```

```
  /*
    array(4) {["河南"]=>string(6)"郑州" [0]=>string(6)"南京"
    [1]=>string(6)"广州" [2]=>string(6)"成都"} int(4)
  */
?>
```

说明：

- array_push()函数向数组 $ city 中添加了两个元素：'广州'和'成都'。两个元素从最后添加，且'广州'在'成都'之前。新数组 $ city 元素个数增加到 4。
- array_push()函数返回值是新数组 $ city 的元素个数，即 4。

6.2.3 数组元素的删除

在 PHP 中要想删除数组元素，可以使用 array_shift()函数、array_pop()函数和 unset()函数。其中，array_shift()函数用于删除数组开头元素，array_pop()函数用于删除数组末尾的元素，而 unset()函数则可以删除指定的数组元素。

1. 使用 array_shift()函数进行数组元素的删除

array_shift()函数将数组开头的单元移出数组。array_shift()函数语法格式为：

```
mixed array_shift ( array &$array )
```

说明：array_shift() 将 array 的第一个单元移出并作为结果返回，将 array 的长度减 1 并将所有其他单元向前移动一位。所有的数字键名将改为从零开始计数，文字键名将不变。array 是要输入的数组，函数返回移出的值。如果 array 为空或不是一个数组，则返回 NULL。

【示例 7】 使用 array_shift()函数进行数组元素的删除。

eg7.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $sanguo =array('Shuguo'=>'Liubei','Weiguo'=>'Caocao','Wuguo'=>'Sunquan');
  $person =array_shift($sanguo);
  echo $person.'<br />';   //输出 Liubei,键忽略,返回的只有值
  var_dump($sanguo);
//得到新数组:array(2){["Weiguo"]=>string(6) "Caocao" ["Wuguo"]=>string(7) "Sunquan" }
?>
```

说明：

- 数组删除一个元素，是头元素，即'Shuguo'=>'Liubei'，返回的是数组元素的值，即'Liubei'。
- 删除元素后的新数组元素的所有数字键名将改为从零开始计数，文字(字符)键名将不变。

2. 使用 array_pop()函数进行数组元素的删除

array_pop()函数用来将数组最后一个单元弹出(出栈)。使用该函数可以用来删除数组元素。array_pop()函数的语法格式为：

```
mixed array_pop ( array &$array )
```

说明：array_pop() 弹出并返回 array 数组的最后一个单元，并将数组 array 的长度减 1。如果 array 为空（或者不是数组），将返回 NULL。

【示例 8】 使用 array_pop() 函数进行数组元素的删除。

eg8.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $sanguo = array('Shuguo'=>'Liubei','Weiguo'=>'Caocao','Wuguo'=>'Sunquan');
  $person = array_pop($sanguo);
  echo $person.'<br />';   //输出为 Sunquan，键忽略，返回的只有值
  var_dump($sanguo);
  //得到新数组：array(2){["Shuguo"]=>string(6)"Liubei" ["Weiguo"]=>string(6)"Caocao" }
?>
```

说明：

- array_pop() 函数删除最后一个元素，数组长度减 1，并返回该数组元素，因此本程序返回 'Sunquan'。
- array_pop() 函数的参数必须是变量名，因此本程序中不能写成 $person = array_pop(array('Shuguo'=>'Liubei','Weiguo'=>'Caocao','Wuguo'=>'Sunquan'))，而只能写成 $person = array_pop($sanguo)。

3. 使用 unset() 函数删除指定的数组元素

在前面已经用到 unset() 函数，在数组中也可以使用 unset() 函数来删除数组元素。此时，可以把数组元素看成普通的 PHP 变量。

【示例 9】 使用 unset() 函数删除指定的数组元素。

eg9.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $sanguo = array('Shuguo'=>'Liubei','Weiguo'=>'Caocao',
    'Wuguo'=>'Sunquan','Guanyu','Zhangfei');
  unset($sanguo['Shuguo']);              //删除 'Liubei'
  unset($sanguo[0],$sanguo['Wuguo']);    //删除 'Guanyu'和 'Sunquan'
  var_dump($sanguo);
  //剩下 array(2) { ["Weiguo"]=>string(6) "Caocao" [1]=>string(8) "Zhangfei" }
?>
```

6.3　数组的遍历

在 PHP 中想了解数组的键和值可以使用 var_dump() 函数，但是一般仅限于在程序的调试阶段。其实在 PHP 中还提供了多种数组的遍历方法，使数组元素的访问变得非常方便。

6.3.1　使用 for 语句遍历数组

在 PHP 中如果数组的索引是整型数，在这种情况下使用 for 语句遍历数组非常方便。

【示例 10】 使用 for 语句遍历数组。

eg10.php 代码如下。

```
<?php
  header('content-type:text/html;charset=utf-8');
  $city=array('北京','上海','广州','深圳','天津','重庆','苏州','武汉');
  for($i=0;$i<count($city);$i++)//count()函数用于统计数组元素个数,读者可自查手册
    echo "\t".$city[$i];
?>
```

【示例 11】 使用 for 语句遍历数组,下面的程序用来求班上学生的语文成绩最高分。

eg11.php 代码如下。

```
<?php
  $a1=array( array('id'=>'01','name'=>'Liubei','Yuwen'=>90),
             array('id'=>'02','name'=>'Guanyu','Yuwen'=>93),
             array('id'=>'03','name'=>'Zhangfei','Yuwen'=>95),
             array('id'=>'04','name'=>'Zhaoyun','Yuwen'=>89),
             array('id'=>'05','name'=>'Machao','Yuwen'=>79),
             array('id'=>'06','name'=>'Huangzhong','Yuwen'=>87),
        );
  $max=$a1[0]['Yuwen'];                    //假设第 1 个人是最高分,记下该生的成绩
  $p=0;                                    //记下该生的 key 值
  for($i=0;$i<count($a1);$i++){            //遍历每一个元素,与最高分比较大小
    if (($a1[$i]['Yuwen'])>$max){          //若某生分数比最高分高
      $max=$a1[$i]['Yuwen'];               //则修改最高分为该生的分数
      $p=$i;                               //且记下他的 key 值
    }
  }
  echo 'Zuigaofen: '.$a1[$p]['id'].'----'.$a1[$p]['name'].'----'.$a1[$p]['Yuwen'];
  //输出最高分学生信息
?>
```

说明:本程序中学生信息保存在数组 $a1 中。而 $a1 的索引(键)是整型数,方便使用 for 语句进行遍历。

count()函数用于计算数组元素的个数,$a1 数组有 6 个元素,因此返回 6。

6.3.2 使用 foreach 语句遍历数组

for 语句在遍历索引为数字的数组时比较方便。如果数组的索引是字符,使用 for 语句遍历数组就非常不方便了。在 PHP 中提供了 foreach 语句,foreach 语法结构提供了遍历数组的简单方式。foreach 仅能够应用于数组和对象,如果尝试应用于其他数据类型的变量,或者未初始化的变量,将发出错误信息。foreach 语句有两种语法格式:

```
foreach (array_expression as $value)
    statement

foreach (array_expression as $key=>$value)
    statement
```

第一种格式遍历给定的 array_expression 数组。每次循环中，当前单元的值被赋给 $value 并且数组内部的指针向前移一步(下一次循环中将会得到下一个单元)。

第二种格式做同样的事，除了当前单元的键名，当前单元的值也会在每次循环中被赋给变量 $key。

提示：当 foreach 开始执行时，数组内部的指针会自动指向第一个单元，这意味着不需要在 foreach 循环之前调用 reset()。由于 foreach 依赖内部数组指针，在循环中修改其值将可能导致意外的行为。

例如：

```
<?php
  $sanguo =array('Shugguo'=>'Liubei','Weiguo'=>'Caocao','Wuguo'=>'Sunquan');
  foreach($sanguo as $v){  //把数组中的每一个元素都当成$v来处理,不考虑索引
    echo $v.'<br />';
  }
?>
```

- 本程序会输出三行数组元素，分别是'Liubei'、'Caocao'和'Sunquan'。
- foreach($sanguo as $v)语句会把数组 $sanguo 中的每一个元素都赋值给 $v 来进行处理，不考虑索引。

如果要考虑索引，代码可以写成：

```
<?php
  $sanguo =array('Shugguo'=>'Liubei','Weiguo'=>'Caocao','Wuguo'=>'Sunquan');
  foreach($sanguo as $k=>$v){
    echo $k.'----'.$v.'<br />';
  }
?>
```

foreach($sanguo as $k=>$v)语句把数组每一个元素的索引和值分别赋值到变量 $k 和 $v 中。as 前面必须是数组名。$k 和 $v 可以是合法的变量名，习惯上使用 $key 和 $value。

foreach 语句能够自动遍历到数组的每一个元素。能够从第 1 个元素开始，自动移动指针到下一个数组元素。

【示例 12】 使用 foreach 语句遍历数组，下面的程序用来求班上学生的语文成绩最高分。

eg12.php 代码如下。

```
<?php
  $a1 =array( array('id'=>'01','name'=>'Liubei','Yuwen'=>90),
              array('id'=>'02','name'=>'Guanyu','Yuwen'=>93),
              array('id'=>'03','name'=>'Zhangfei','Yuwen'=>95),
              array('id'=>'04','name'=>'Zhaoyun','Yuwen'=>89),
              array('id'=>'05','name'=>'Machao','Yuwen'=>79),
              array('id'=>'06','name'=>'Huangzhong','Yuwen'=>87),
       );
  $max =$a1[0]['Yuwen'];
  $p =0;
```

```
    foreach($a1 as $k=>$v){
      if ($v['Yuwen']>$max){
        $p=$k;
        $max=$v['Yuwen'];
      }
    }
    echo 'Zuiganfen: '.$a1[$p]['id'].'----'.$a1[$p]['name'].'----'.$a1[$p]['Yuwen'];
?>
```

说明：

- 本例中可以使用 foreach 遍历数组 $a1，而 $a1 的每一个元素又是一个数组(一维数组)。因此在执行 foreach($a1 as $k=>$v)语句时得到的每一个 $v 是一个一维数组。可以通过一维数组的索引来访问该一维数组的'Yuwen'索引所对应的值，即 $v['Yuwen']；$k 是 $a1 的第一个索引，即它的值是 0～5。
- 求最高分的思路与示例 11 相同。

6.3.3　使用 list()遍历数组

在 PHP 中提供了一个名为 list()的语言结构，该语法结构用于将数组中的一些元素赋值给一些变量。list()的语法格式为：

```
array list ( mixed $varname [, mixed $ ...] )
```

严格地说，像 array()一样，这不是真正的函数，而是语言结构。list()用一步操作给一组变量进行赋值。list()能够返回指定的数组。

list()仅能用于数字索引的数组并假定数字索引从 0 开始。

例如：

```
<?php
  $person=array('Shuguo'=>'Liubei','Guanyu','Zhangfei','Weiguo'=>'Caocao');
  list($a1,$a2)=$person;  //得到'Guanyu'和'Zhangfei',取出索引为 0 和 1 的元素
  echo $a1,$a2;           //输出'Guanyu'和'Zhangfei'
?>
```

再如：

```
<?php
  $person=array('Shuguo'=>'Liubei','Guanyu','Zhangfei','Weiguo'=>'Caocao','Machao');
  list($a1,,$a3)=$person; //得到'Guanyu'和'Machao',取出索引为 0 和 2 的元素
  echo $a1,$a3;           //输出'Guanyu'和'Machao'
?>
```

【示例 13】 使用 list 语言结构遍历数组。

eg13.php 代码如下。

```
<?php
  $a1=array( array('id'=>'01','name'=>'Liubei','Yuwen'=>90),
             93,
             array('id'=>'03','name'=>'Sunquan','Yuwen'=>95),
       );
```

```
  list($liubei,$caocao,$sunquan)=$a1;
  var_dump($liubei); echo '<br />';   //得到'Liubei'的信息,是一个数组
  var_dump($caocao); echo '<br />';   //得到 93
  var_dump($sunquan);                 //得到 Sunquan 的信息,是一个数组
?>
```

6.3.4　使用 each()函数遍历数组

在 PHP 中提供了一个名为 each()的函数,使用该函数可以得到当前数组元素的 key 和 value,并把指向数组的下一个元素,若达到了数组的尾部则返回 false。如果确定要返回哪个元素,可以直接引用。each()函数的语法格式为:

```
array each ( array &$array )
```

返回数组中当前的键/值对并将数组指针向前移动一步,在执行 each()之后,数组指针将停留在数组中的下一个单元或者当碰到数组结尾时停留在最后一个单元。如果要再用 each 遍历数组,必须使用 reset()函数将数组指针重置。

【示例 14】　使用 each()函数遍历数组。

eg14.php 代码如下。

```
<?php
  header('content-type:text/html;charset=utf-8');
  $citylist =array('首都'=>'北京','四川'=>'成都','湖北'=>'武汉','西安','南京','浙江'
  =>'杭州');
  $a1 =each($citylist);
  echo $a1['key'].'---'.$a1['value'].'<br />';     //输出为"首都---北京",并指向'四川'=
                                                    >'成都'
  $a2 =each($citylist);
  echo $a2['key'].'---'.$a2['value'].'<br />';     //输出为"四川---成都",并指向'湖北'=
                                                    >'武汉'
  while($a3 =each($citylist)){
     echo $a3['key'].'---'.$a3['value'].'<br />';  //从"'湖北'=>'武汉'"开始
  }
  reset($citylist);                                  //重置数组指针,指向"'首都'=>'北京'"
  while($a4 =each($citylist)){
     echo $a4['key'].'---'.$a4['value'].'<br />';  //从"'首都'=>'北京'"开始输出
  }
?>
```

6.4　数 组 排 序

排序是计算机学科中一个重要问题,在 C 语言、数据结构等课程中,读者学过一些排序,例如冒泡排序、选择法排序、堆排序、希尔排序,等等。在 PHP 中提供了一些系统函数,使用它们可以非常方便地实现数组元素的排序。

6.4.1　sort()、rsort()、ksort()和 krsort()函数

1. 使用 sort()函数排序

在 PHP 中可以使用 sort()函数对数组进行排序。当本函数结束时,数组单元将被从最低到最高重新安排。sort()函数删除原来数组的索引,sort()函数对原数组进行排序,因此不会返回数组(成功返回 true,失败返回 false)。sort()数组的语法格式为:

```
bool sort ( array &$ array [, int $ sort_flags = SORT_REGULAR ] )
```

本函数对数组进行排序。当本函数结束时数组单元将被从最低到最高重新安排。array 是要排序的数组,sort_flags 是可选的第二个参数,可以用以下值改变排序的行为。

- SORT_REGULAR:正常比较单元(不改变类型)。
- SORT_NUMERIC:单元被作为数字来比较。
- SORT_STRING:单元被作为字符串来比较。
- SORT_LOCALE_STRING:根据当前的区域(locale)设置来把单元当作字符串比较,可以用 setlocale()来改变。
- SORT_NATURAL:与 natsort()类似,对每个单元以"自然的顺序"对字符串进行排序。这是 PHP 5.4.0 中新增的。
- SORT_FLAG_CASE:能够与 SORT_STRING 或 SORT_NATURAL 合并(OR 位运算),不区分大小写排序字符串。

排序成功则返回 true,排序失败则返回 false。

【示例 15】 使用 sort()函数进行数组排序。

eg15.php 代码如下。

```
<?php
  $fruits  =array("lemon","Orange","banana","apple");
  sort($fruits );
  foreach($fruits as $key=>$val){
    echo "fruits[".$key."]=".$val."<br />" ;
  }
  /* 运行结果为:
  fruits[0]=Orange
  fruits[1]=apple
  fruits[2]=banana
  fruits[3]=lemon
  */
  $fruits  =array("lemon","Orange","banana","apple");
  sort($fruits,SORT_NATURAL | SORT_FLAG_CASE );     //不区分大小写自然排序
  foreach($fruits as $key=>$val){
    echo "fruits[".$key."]=".$val."<br />" ;
  }
  /* 运行结果为:
  fruits[0]=apple
  fruits[1]=banana
  fruits[2]=lemon
  fruits[3]=Orange
  */
```

```
?>
```

2. 使用 rsort()函数排序

rsort()函数和 sort()函数类似，不同之处在于前者是升序排列而后者是降序排列。rsort()函数的语法格式为：

```
bool rsort ( array &$array [, int $sort_flags = SORT_REGULAR ] )
```

本函数对数组进行逆向排序(最高到最低)。本函数对数组进行排序，当本函数结束时数组单元将被从最高到最低重新安排。array 是要排序的数组，sort_flags 是可选的第二个参数，具体含义请参照 sort()函数。

【示例 16】 使用 rsort()函数进行数组排序。

eg16. php 代码如下。

```
<?php
  $fruits  =array("lemon","Orange","banana","apple");
  sort($fruits );
  foreach($fruits as $key=>$val){
    echo "fruits[".$key."]=".$val."<br />" ;
  }
  /* 运行结果为:
  fruits[0]=Orange
  fruits[1]=apple
  fruits[2]=banana
  fruits[3]=lemon
  */
  $fruits  =array("lemon","Orange","banana","apple");
  rsort($fruits,SORT_NATURAL | SORT_FLAG_CASE );    //不区分大小写自然排序
  foreach($fruits as $key=>$val){
    echo "fruits[".$key."]=".$val."<br />" ;
  }
  /*
  fruits[0]=Orange
  fruits[1]=lemon
  fruits[2]=banana
  fruits[3]=apple
  */
?>
```

3. 使用 ksort()函数和 krsort()函数排序

在 PHP 中 sort()函数和 rsort()函数排序是根据值来进行的。如果需要根据索引(键)排序则可以使用另外两个函数，即 ksort()函数和 krsort()函数。ksort()函数用于按照键名升序排列，它的语法格式为：

```
bool ksort ( array &$array [, int $sort_flags = SORT_REGULAR ] )
```

对数组按照键名升序排序，保留键名到数据的关联。本函数主要用于关联数组。array 是输入的数组，可以用可选参数 sort_flags 改变排序的行为，详情见 sort()。

krsort()函数用于按照键名降序排列。krsort()函数的语法格式为：

```
bool krsort ( array &$ array [, int $ sort_flags = SORT_REGULAR ] )
```

对数组按照键名逆向排序，保留键名到数据的关联，主要用于结合数组。rray 是输入的数组，可以用可选参数 sort_flags 改变排序的行为，详情见 sort()。

【示例 17】 使用 ksort()函数和 krsort()函数进行数组索引排序。

eg17.php 代码如下。

```
<?php
  $citylist = array('shoudu'=>'Beijing','sichuan'=>'Chengdu',
  'hubei'=>'Wuhan','zhejiang'=>'Hangzhou');
  ksort($citylist);                          //按索引排序(升序),不打乱索引和值的对应性
  foreach($citylist as $key=>$value){
    echo $key.'---- '.$value.'<br />';
  }
  /*
  hubei----Wuhan
  shoudu----Beijing
  sichuan----Chengdu
  zhejiang----Hangzhou
  */
  $citylist = array('shoudu'=>'Beijing','sichuan'=>'Chengdu',
  'hubei'=>'Wuhan','zhejiang'=>'Hangzhou');
  krsort($citylist);                         //按索引排序(降序),不打乱索引和值的对应性
  foreach($citylist as $key=>$value){
    echo $key.'---- '.$value.'<br />';
  }
  /*
  zhejiang----Hangzhou
  sichuan----Chengdu
  shoudu----Beijing
  hubei----Wuhan
  */
?>
```

6.4.2 使用 shuffle()函数进行随机排序

在 PHP 中可以使用 shuffle()函数对数组进行随机排序。随机排序就是将原有的数组随机打乱，因此并不会产生新的数组。shuffle()函数的语法格式为：

```
bool shuffle ( array &$ array )
```

本函数打乱(随机排列单元的顺序)一个数组。array 就是要操作的数组，只能是变量形式。随机排序成功则返回 true，随机排序失败则返回 false。

【示例 18】 使用 shuffle()函数对数组进行随机排序。

eg18.php 代码如下。

```
<?php
  header('content-type:text/html;charset=utf-8');
  $a1 = range(1,10,2);
```

```
    echo '排序前:<br />';
    foreach($a1 as $value){
      echo "\t".$value;
    }
    /*
    排序前:
    1  3  5  7  9
    */
    shuffle($a1);
    echo '<br />排序后:<br />';
    foreach($a1 as $value){
      echo "\t".$value;
    }
    /*
    排序后:
    7  3  9  5  1
    */
?>
```

6.4.3　使用 array_reverse()函数进行反向排序

在 PHP 中可以使用函数 array_reverse()对数组进行反向排序,产生新的数组,原来的数组保持不变。array_reverse()函数的语法格式为:

```
array array_reverse ( array $array [, bool $preserve_keys = false ] )
```

array_reverse()接受数组 array 作为输入并返回一个单元为相反顺序的新数组。array 是要输入的数组,preserve_keys 设置为 true,则保留数字键,否则不保留数字键。非数字的键则不受这个设置的影响,总是会被保留。返回值是反转后的新数组。

【示例 19】　使用 array_reverse() 函数对数组进行反向排序。

eg18. php 代码如下。

```
<?php
  $input = array("php","mysql",array("green","red"));
  $result = array_reverse($input);                //反向排序,不保留数字键
  $result_keyed = array_reverse($input,true);     //反向排序,保留原来的数字键
  var_dump($input);
  /*
  输出为 array(3){[0]=>string(3)"php" [1]=>string(5)"mysql"
  [2]=>array(2){[0]=>string(5)"green" [1]=>string(3) "red"}}
  */
  echo '<br />';
  var_dump($result);
  /*
  输出为 array(3){[0]=>array(2){[0]=>string(5)"green" [1]=>string(3)"red"}
  [1]=>string(5)"mysql" [2]=>string(3)"php"}
  */
  echo '<br />';
  var_dump($result_keyed);
  /*
```

```
  输出为 array(3){[2]=>array(2){[0]=>string(5)"green" [1]=>string(3)"red"}
  [1]=>string(5)"mysql" [0]=>string(3)"php"}
   */
?>
```

说明：

- array_reverse()函数对数组进行排序时，若将第 2 个参数 $preserve_keys 设置为 true，则表示保留原来的数字键；若设置为 false，则不保留原来的数字键。
- $preserve_keys 的默认值为 false，表示不保留数字键。无论 $preserve_keys 设置为何值，数字的字符键都不会受到影响。
- array_reverse()函数执行完毕产生新数组，原来的数组不受影响。

6.5 数组的其他操作

前面几节中讲述了数组的添加、删除、排序等操作。在 PHP 中还提供了一些函数用于对数组进行其他操作，例如随机获取数组元素、联合数组、合并数组、拆分数组、替换数据和判断数组的类型等。

6.5.1 随机获取数组元素

在 PHP 中提供了一个名为 array_rand()的函数，该函数能够从数组中随机抽取一个或多个随机元素，使用它非常有用，比如可以用在题库中随机抽取若干题目。array_rand()函数的语法格式为：

```
mixed array_rand ( array $input [, int $num_req =1 ] )
```

函数的功能是从数组中取出一个或多个随机的单元，并返回随机条目的一个或多个键。input 是输入的数组。num_req 指明了要取出多少个单元，默认值为 1。如果指定的数目超过了数组里的数量，将会产生一个 E_WARNING 级别的错误。

返回值：如果只取出一个，array_rand()函数返回一个随机单元的键名，否则就返回一个包含随机键名的数组。这样就可以随机从数组中取出键名和值。

【示例 20】 使用 array_rand()函数从数组中随机抽取元素。

eg20.php 代码如下。

```
<?php
  $citylist =array('shoudu'=>'Beijing','sichuan'=>'Chengdu',
  'hubei'=>'Wuhan','zhejiang'=>'Hangzhou');
  $a =array_rand($citylist);    //$a 是一个字符串，它是$citylist 的键
  echo $citylist[$a].'<br />';
  $b =array_rand($citylist,3);  //$b 是一个数组，数组的值是$citylist 的键
  echo $citylist[$b[0]].'---'.$citylist[$b[1]].'---'.$citylist[$b[2]];
?>
```

程序运行结果如图 6-1 所示。

说明：



图 6-1　随机从数组中抽取元素

- 若从数组随机抽取一个元素，得到的将是该数组的某一个键名(随机的)，再配合数组名即可得到该数组元素。上例中得到的随机键名是'zhejiang'，再配合数组名可以得到的值为'Hangzhou'。
- 若从数组中取出一些元素，得到的将是该数组的一些键名(随机的，但是先后顺序没有变化)，然后再配合数组名即可得到这些数组元素。得到的这些键名是一个数组，上例中得到的数组是 array('shoudu'、'sichuan'、'hubei')，配合数组名得到的是'Beijing'、'Chengdu'、'Wuhan'。

【示例 21】 使用 array_rand()函数从若干个题目中随机抽取几道题目并显示出来。

eg21.php 代码如下。

```
<?php
  header('content-type:text/html;charset=utf-8');
  $question=array(
              array('中国最大的城市是?','上海','天津','广州','重庆','D'),
              array('千湖之省是?','湖南','湖北','四川','浙江','B'),
              array('CPU是','存储器','中央处理器','总线','显示器','B'),
              array('面积最大的省份是?','西藏','黑龙江','新疆','青海','C'),
              array('GDP排第一的省份是?','广东','江苏','浙江','山东','A')
           );
  $test=array_rand($question,3);
  $i=0;
  foreach($test as $value){
    $i++;
    echo "题目$i. " .$question[$value][0].'<br />';
    echo " A.".$question[$value][1].'<br />';
    echo " B.".$question[$value][2].'<br />';
    echo " C.".$question[$value][3].'<br />';
    echo " D.".$question[$value][4].'<br />';
    echo "<br />";
  }
?>
```

程序运行结果如图 6-2 所示。

说明：

- 本例中，$test 是一个数组，它有三个元素(分别是 $test[0]、$test[1]和 $test[2])，每一个元素都是数组 $question 的一个键，也就是说 $question[$test[0]]、$question[$test[1]]、$question[$test[2]]分别表示 3 个题目信息。

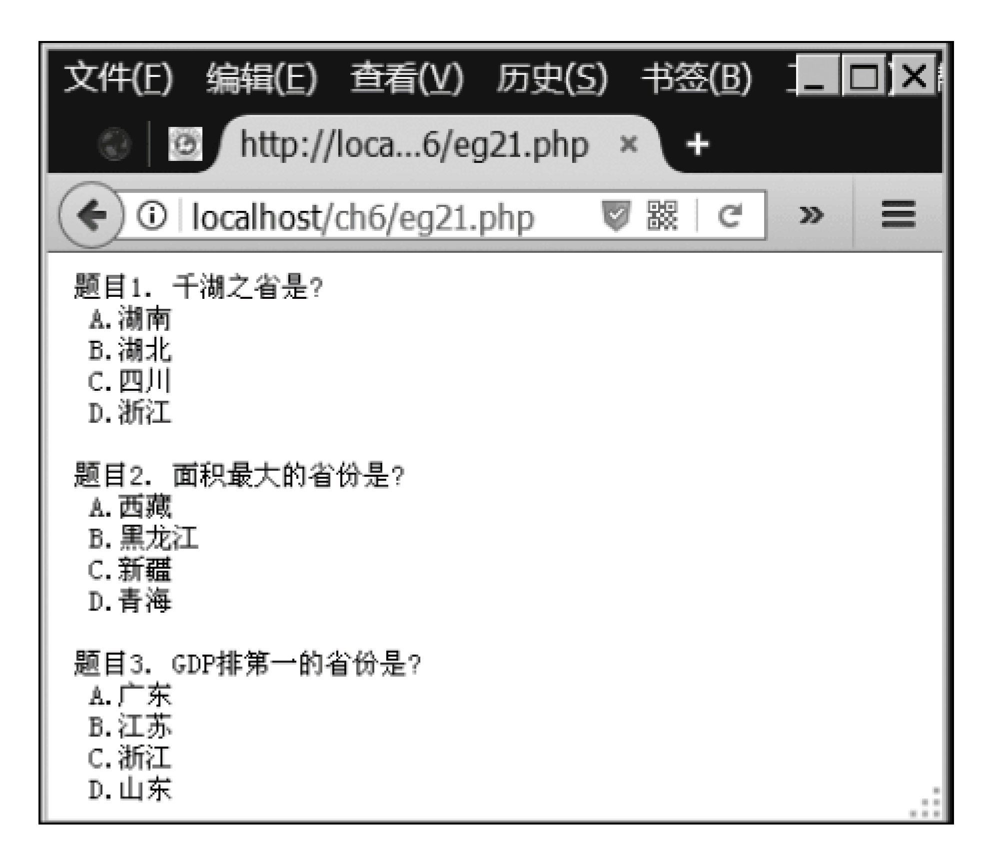


图 6-2 随机抽取题目

- 而 $question[$test[0]][0]、$question[$test[0]][1]、$question[$test[0]][2]、$question[$test[0]][3]、$question[$test[0]][4]则表示第一个题目的详细信息。

6.5.2 联合数组

所谓联合数组，就是将两个数组联合起来，一个数组的值作为元素的键名，另一个数组的值作为元素的值。在 PHP 中使用 array_combine()函数实现数组的联合，该函数的语法格式为：

```
array array_combine ( array $keys, array $values )
```

返回一个 array，用来自 keys 数组的值作为键名，来自 values 数组的值作为相应的值。如果两个数组元素的个数不相同，则返回 false，且会抛出异常。

【示例 22】 使用 array_combine()函数实现数组的联合。

eg22.php 代码如下。

```
<?php
  $k1 =array('shoudu','sichuan','hubei','zhejiang');
  $v1 =array('Beijing','Chengdu','Wuhan','Hangzhou');
  $citylist1 =array_combine($k1,$v1);
  var_dump($citylist1);
  /*
  array(4){["shoudu"]=>string(7)"Beijing" ["sichuan"]=>string(7)"Chengdu"
  ["hubei"]=>string(5)"Wuhan" ["zhejiang"]=>string(8)"Hangzhou"}
   */
  echo '<br />';
  $k2 =array('shoudu','sichuan','hubei');
  $v2 =array('Beijing','Chengdu','Wuhan','Hangzhou');
  //$k2 和$v2 二者元素个数不相等，若联合则产生异常，并返回 false
  $citylist2 =array_combine($k2,$v2);
  var_dump($citylist2);
?>
```

6.5.3 合并数组

在 PHP 中数组的合并方式有三种：第 1 种是使用 array_merge()函数进行合并，第 2 种是使用 array_merge_recursive()函数进行合并，第 3 种是使用加号“+”进行合并。

1. 使用 array_merge()函数合并数组

array_merge()函数用于合并一个或多个数组，它会将一个数组的值附加在另一个数组的后面，并返回新数组。array_merge()函数的语法格式为：

```
array array_merge ( array $array1 [, array $ ... ] )
```

array_merge()函数将一个或多个数组的单元合并起来，一个数组中的值附加在前一个数组的后面，返回作为结果的数组。如果输入的数组中有相同的字符串键名，则该键名后面的值将覆盖前一个值。然而，如果数组包含数字键名，后面的值将不会覆盖原来的值，而是附加到后面。如果只给了一个数组并且该数组是数字索引的，则键名会以连续方式重新索引。$array1 是合并之后在最前面的数组，后面的数组依次合并在后。合并产生新数组，原来的数组保持不变。

【示例 23】 使用 array_merge ()函数实现数组的合并。

eg23.php 代码如下。

```
<?php
  $array1 =array("color" =>"red",2,4);
  $array2 =array("a","b","color"=>"green","shape"=>"trapezoid",4);
  $result =array_merge($array1,$array2);
  print_r($result);
  echo '<br />';
  print_r($array1);
  echo '<br />';
  print_r($array2);
  /* 运行结果为：
  array([color]=>green [0]=>2 [1]=>4 [2]=>a [3]=>b [shape]=>trapezoid [4]=>4)
  array([color]=>red [0]=>2 [1]=>4)
  array([0]=>a [1]=>b [color]=>green [shape]=>trapezoid [2]=>4)
   */
?>
```

说明：

- 本例中，使用 array_merge()函数合并数组，原数组 array1 和 array2 都没有改变，生成了新数组 $result。
- 新数组中，$array1 在前面，$array2 在后面。若有相同的字符串键名，则该键名后面的值将覆盖前一个值，这里"color"=>"green"覆盖"color" => "red"。
- 数字键名相同则不覆盖，因此 0=>"a"没有覆盖 0=>2，而是重新索引。

2. 使用 array_merge_recursive()函数合并数组

array_merge_recursive()函数递归合并一个或多个数组元素，一个数组中的值附加在另一个数组的后面，并返回一个新数组。array_merge_recursive()函数的语法格式为：

```
array array_merge_recursive ( array $array1 [, array $ ...] )
```

array_merge_recursive()函数将一个或多个数组的单元合并起来，一个数组中的值附加在前一个数组的后面。返回作为结果的数组。如果输入的数组中有相同的字符串键名，则这些值会被合并到一个数组中，还将递归下去。如果一个值本身是一个数组，该函数将按照相应的条目把它合并为另一个数组。但是，如果数组具有相同的数字键名，后一个值将不会覆盖原来的值，而是附加到后面。参数 array1 是要合并的初始数组。省略的内容是数组变量列表，进行递归合并。该函数的返回值是一个结果数组，其中的值合并自附加的参数。

【示例 24】 使用 array_merge_recursive()函数实现数组的合并。

eg24. php 代码如下。

```php
<?php
  echo "<pre><font size='-1'>";
  $a1 =array('first'=>'bob','last'=>'jones' ,'age'=>'48','male');
  $a2 =array( 'last'=>'smith' ,'age'=>'41','male');
  $a3 =array('first'=>'pete','last'=>null,'age'=>'3');
  $a4 =array('first'=>'joe','last'=>'johnson','age'=>'33','female');
  $a5 =array_merge_recursive($a1,$a2,$a3,$a4);
  print_r($a5);
  echo "</font></pre>";
?>
```

程序运行结果如图 6-3 所示。

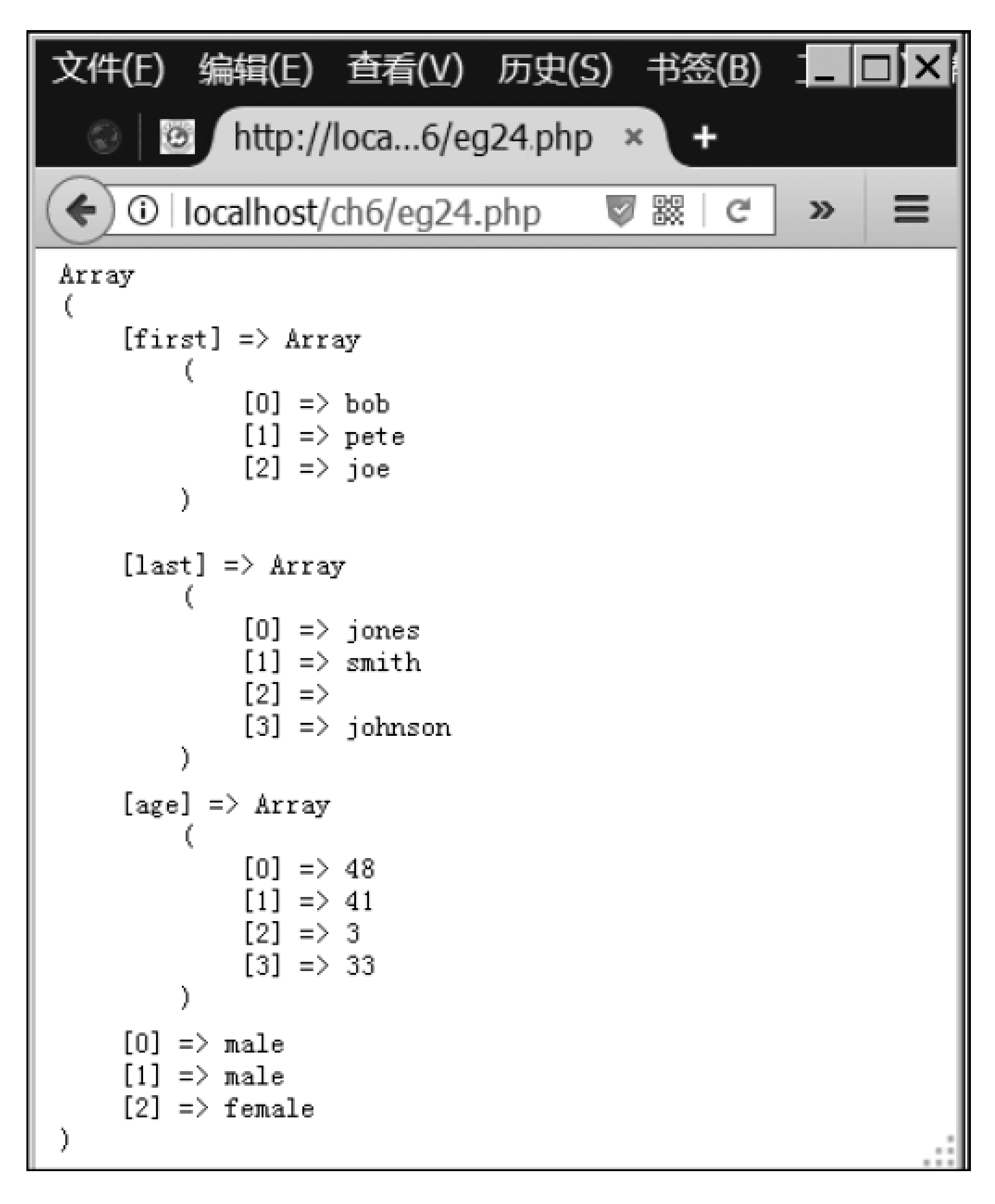


图 6-3 使用 array_merge_recursive()函数实现数组的合并

说明：

• 本例中，$a1、$a3 和 $a4 这 3 个数组中有字符串键名 first，则这 3 个元素合并成

一个新数组，这个新数组是结果数组的一个元素，该元素在结果数组中键名为 first；同理字符串键 last 和 age 也是一样。

- 而 $ a1 中的 male、$ a2 中的 male 和 $ a4 中的 female 则是数字键名，即使它们键名相同也不合并为新数组，而是直接附加到结果数组的后面。

3. 使用加号"＋"对数组进行合并

使用加号操作符"＋"合并数组是一种最简单的数组合并方式，对两个数组或多个数组进行合并时，键名和值都保持原样，是原封不动地进行合并的。如果两个数组中的键名是相同的，那么以第 1 个数组的值为主，即第 2 个加号"＋"后面的数组的值会自动忽略。

【示例 25】 使用加号"＋"实现数组的合并。

eg25. php 代码如下。

```
<?php
  echo "<pre><font size='-1'>";
  $a1 =array('Hubei'=>'Wuhan','Hunan'=>'Changsha','Zhejiang'=>'Hangzhou','Guangzhou');
  $a2 =array('Hubei'=>'Yichang','Hunan'=>'Yueyang','Jiangxi'=>'Nanchang','Shenzhen','
  Foshan');
  $result =$a1 +$a2;
  print_r($result);
  echo "</font></pre>"
?>
```

程序运行结果如图 6-4 所示。



图 6-4　使用加号"＋"实现数组的合并

说明：

- 本例中，使用加号"＋"实现数组的合并，以 $ a1 为主，$ a1 中的每一个元素都会保留下来。
- 在 $ a2 中键名相同的元素被忽略，键名不相同的元素则保留。比如 Jiangxi'＝＞'Nanchang' 和'Foshan'全部保留，而'Hubei'＝＞'Yichang'、'Hunan'＝＞'Yueyang'和'Shenzhen' 因键名和 $ a1 中元素相同被忽略。

6.5.4　拆分数组

在 PHP 中提供了一个名为 array_slice() 的函数用于数组的拆分。该函数的语法格式为：

```
array array_slice(array $array,int $offset [,int $length=NULL [,bool $preserve_keys=false
]])
```

array_slice()返回根据 offset 和 length 参数所指定的 array 数组中的一段序列(数组)。有关参数说明如下。

- array 参数：这是一个输入的数组。
- offset 参数：如果 offset 为非负值,则序列将从 array 中的此偏移量开始。如果 offset 为负值,则序列将从 array 中距离末端最远的地方开始。
- length 参数：如果给出了 length 并且为正值,则序列中将具有给定数量的单元。如果给出了 length 并且为负值,则序列将终止在距离数组末端的地方。如果省略,则序列将从 offset 开始一直到 array 的末端。
- preserve_keys 参数：注意 array_slice()默认会重新排序并重置数组的数字索引。可以通过将 preserve_keys 设为 true 来改变此行为。

【示例 26】 使用 array_slice()函数拆分数组。

eg26.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $sanguo =array('刘备','关羽','张飞','赵云','马超','黄忠','曹操','曹仁','曹丕','曹植');
  $a =array_slice($sanguo,1,5);      //1~5,即关羽到黄忠 5个人,刘备对应 0
  $b =array_slice($sanguo,6);        //6到最后,即曹操、曹仁、曹丕、曹植
  print_r($a);
  print_r($b);
  $c =array_slice($sanguo,1,-4);     //1到倒数第 5,即关羽到黄忠 5个人
  $d =array_slice($sanguo,6,-1);     //6到倒数第 2,即曹操、曹仁、曹丕
  print_r($c);
  print_r($d);
  $e =array_slice($sanguo,-4);       //最后 4个,即曹操、曹仁、曹丕、曹植
  print_r($e);
  $f =array('刘备','关羽','张飞','魏'=>'曹操','吴'=>'孙权');
  $g =array_slice($f,1,3,true);      //不重置键名
  $h =array_slice($f,1,4,false);     //重置键名,此为默认状态
  print_r($g);                       //array([1] =>关羽,[2]=>张飞,[魏]=>曹操)
  print_r($h);                       //array([0] =>关羽,[1]=>张飞,[魏]=>曹操,[吴]=>孙
                                       权)
?>
```

6.5.5 替换数组

在 PHP 中提供了名为 array_splice()函数,可以把数组中的一部分去掉并用其他值取代。该函数的语法格式为：

```
array array_splice ( array &$input, int $offset [, int $length =0 [, mixed $replacement ]])
```

把 input 数组中由 offset 和 length 指定的单元去掉,如果提供了 replacement 参数,则用其中的单元取代。注意 input 中的数字键名不被保留。返回值为一个包含有被移除单元

的数组。

有关参数说明如下。

- input：输入的数组。
- offset：如果 offset 为正值，则从 input 数组中该值指定的偏移量开始移除；如果 offset 为负值，则从 input 末尾倒数该值指定的偏移量开始移除。
- length：如果省略 length，则移除数组中从 offset 到结尾的所有部分；如果指定了 length 并且为正值，则移除这么多单元；如果指定了 length 并且为负值，则移除从 offset 到数组末尾倒数 length 为止中间所有的单元。小窍门：当给出了 replacement 并要移除从 offset 到数组末尾所有单元时，用 count($input)作为 length。
- replacement：如果给出了 replacement 数组，则被移除的单元被此数组中的单元替代；如果 offset 和 length 的组合结果是不会移除任何值，则 replacement 数组中的单元将被插入 offset 指定的位置。注意替换数组中的键名不保留。如果用来替换 replacement 只有一个单元，那么不需要给它加上 array()，除非该单元本身就是一个数组、一个对象或者 NULL。

【示例 27】 使用 array_splice()函数替换数组。

eg27. php 代码如下。

```
<?php
  $input =array("red","green","blue","yellow");
  $x =array_splice($input,2);
  //$input 现在是 array("red","green")
  //$x 现在是 array("blue","yellow")
  $input =array("red","green","blue","yellow");
  $x=array_splice($input,1,-1);
  //$input 现在是 array("red","yellow")
  //$x 现在是 array("green","blue")
  $input =array("red","green","blue","yellow");
  $x=array_splice($input,1,count($input),"orange");
  //$input 现在是 array("red","orange")
  //$x 现在是 array("green","blue","yellow")
  $input =array("red","green","blue","yellow");
  $x=array_splice($input,-1,1,array("black","maroon"));
  //$input 现在是 array("red","green","blue","black","maroon")
  //$x 现在是 array("yellow")
  $input =array("red","green","blue","yellow");
  $x=array_splice($input,3,0,"purple");
  //$input 现在是 array("red","green","blue","purple","yellow");
  //$x 现在是 array(0){ }
?>
```

6.5.6　查找键名是否存在

在 PHP 中提供了一个名为 array_key_exists()的函数，用于查找和检测数组中是否有指定的键名。该函数的语法格式为：

```
bool array_key_exists ( mixed $key, array $search )
```

array_key_exists()函数在给定的 key 存在于数组中时返回 true。key 可以是任何能作为数组索引的值。array_key_exists()函数也可用于对象。

有关参数说明如下。

- key：要检查的键。
- search：一个数组，包含待检查的键。

返回值：成功时返回 true，或者在失败时返回 false。

【示例 28】 使用 array_key_exists()函数查找数组的键。

eg28.php 代码如下。

```
<?php
  $search_array = array('first' => 1,'second'=> 4);
  array_key_exists('first',$search_array);           //返回值为 true
  $search_array = array('first' => null,'second'=> 4);
  isset($search_array['first']);                      //返回值为 false
  array_key_exists ('first',$search_array);          //返回值为 true
?>
```

6.5.7 查找值是否存在

在 PHP 中提供了一个名为 in_array()的函数，用于查找数组中是否存在某个值。该函数的语法格式为：

```
bool in_array ( mixed $needle, array $haystack [, bool $strict = false] )
```

在 haystack 中搜索 needle，如果没有设置 strict，则使用宽松的比较。

有关参数说明如下。

- needle：待搜索的值。注意，如果 needle 是字符串，则比较时区分大小写。
- haystack：要查找的数组。
- strict：如果第三个参数 strict 的值为 true，则 in_array()函数还会检查 needle 的类型是否和 haystack 中的相同。

返回值：如果找到 needle 则返回 true，否则返回 false。

【示例 29】 使用 in_array()函数查找某值是否存在于数组中。

eg29.php 代码如下。

```
<?php
  echo '<pre>';
  $os = array("Mac","NT","Irix","Linux");
  if (in_array("Irix",$os)){
    echo "Got Irix\n" ;                               //有输出
  }
  if (in_array("mac",$os)){
    echo "Got mac\n";                         //字符串需比较大小写，不输出
  }
  $a = array('1.10',12.4,1.13);
  if (in_array('12.4',$a,true)){              //第 3 个参数为 true，对数据类型有严格要求
    echo "'12.4' found with strict check\n";  //数据类型不同，不输出
```

```
  }
  if (in_array (1.13,$a,true)){
    echo "1.13 found with strict check\n" ;     //数据类型相同,有输出
  }
  $a =array(array('p','h'),array('p','r'),'o');
  if (in_array(array('p','h'),$a)){             //数组存在
    echo "'ph' was found\n" ;                   //有输出
  }
  if (in_array(array('f','i'),$a)){
    echo "'fi' was found\n" ;                   //不输出
  }
  if (in_array('o',$a)){
    echo "'o' was found\n" ;                    //有输出
  }
?>
```

6.5.8　去掉重复元素值

当数组中出现重复元素时,可以使用 array_unique()函数清理重复的值,该函数的语法格式为:

```
array array_unique ( array $array [, int $sort_flags =SORT_STRING ] )
```

array_unique()函数将 array 作为输入并返回没有重复值的新数组。注意键名保持不变。array_unique()先将值作为字符串排序,然后对每个值只保留第一个遇到的键名,接着忽略所有后面的键名。这并不意味着在未排序的 array 中同一个值的第一个出现的键名会被保留。

有关参数说明如下。

- array:输入的数组。
- sort_flags:第 2 个参数是可选项,可能是下面的一些值。
  - SORT_STRING:默认值。把项目作为字符串来比较。
  - SORT_REGULAR:把每一项按常规顺序排列(Standard ASCII,不改变类型)。
  - SORT_NUMERIC:把每一项作为数字来处理。
  - SORT_LOCALE_STRING:把每一项作为字符串来处理,基于当前区域设置(可通过 setlocale()函数进行更改)。

返回值:返回过滤后的数组。

【示例 30】　使用 array_unique()函数去掉数组中重复的值。

eg30.php 代码如下。

```
<?php
  $input =array("a"=>"green","red","b"=>"green","blue","red");
  $result =array_unique($input);
  var_dump($result);
  //array(3){["a"]=>string(5)"green" [0]=>string(3)"red" [1]=>string(4)"blue" }
  echo '<br />';
  $input =array(4,"4","3",4,3,"3");
```

```
  $result = array_unique($input);
  var_dump($result);
  //array(2){[0]=>int(4) [2]=>string(1)"3" }
?>
```

说明：

- "a"=>"green"和"b"=>"green"二者重复，去掉第 2 个元素；0=>"red"和 1=>"red" 重复，去掉第 2 个，"blue"唯一保留下来。
- 0=>4、1=>"4"、3=>"4"三者重复，保留第 1 个；2=>"3"、4=>3、5=>"3"三者重复，保留第 1 个。

6.5.9 数组的键名和值调换

在 PHP 中提供了 array_flip()函数，该函数使数组的键名和对应的值调换，即键名变成值，而值变成键名。这个数组可以是数字索引数组，也可以是字符索引数组。array_flip()函数一种格式为：

```
array array_flip ( array $trans )
```

array_flip()函数返回一个反转后的 array，例如 trans 中的键名变成了值，而 trans 中的值成了键名。注意 trans 中的值需要能够作为合法的键名，例如需要是 integer 或者 string。如果值的类型不对，将发出一个警告，并且有问题的键/值对将不会反转，直接丢弃。如果同一个值出现了多次，则最后一个键名将作为它的值，所有其他的都丢失了。

有关参数说明如下。

trans 表示要交换键/值对的数组。

返回值：成功时返回交换后的数组，如果失败则返回 NULL。

【示例 31】 使用 array_flip()函数对数组的键和值进行调换。

eg31.php 代码如下。

```
<?php
  $trans = array("a"=>1,"b"=>1,"c"=>2,"d"=>3.4);
  $trans = array_flip($trans);
  print_r($trans);            //array ([1] =>b [2] =>c )
?>
```

说明：

- "a"=>1 与"b"=>1 二者值相同，同一个值出现多次后发生作用，因此调换结果为 1=>"b"。
- "d"=>3.4 中，3.4 不能做键名，产生警告信息，直接丢弃该元素。

6.6 综合案例——考生信息处理

本节使用前面学到的排序知识和数组遍历知识对一个二维数组进行排序。某班有 10 名学生，在一次期中考试中成绩和考生信息如表 6-1 所示。

表 6-1　学生成绩表

学　号	姓　名	分　数	性　别	学　号	姓　名	分　数	性　别
1	Liubei	79	male	6	Lvbu	73	male
2	Guanyu	75	male	7	Caocao	99	male
3	Zhangfei	89	male	8	Sunshangxiang	79	female
4	Diaochan	82	female	9	Dianwei	87	male
5	Machao	76	male	10	Sunce	56	male

要求按照考生从高分到低分输出考生信息。

步骤 1　为了便于编写程序处理，构造考生信息二维数组，代码如下。

```
$stu_info =array(
    array('id'=>1,'name'=>'Liubei','score'=>79 ,'sex'=>'male'),
    array('id'=>2,'name'=>'Guanyu','score'=>75 ,'sex'=>'male'),
    array('id'=>3,'name'=>'Zhangfei','score'=>89 ,'sex'=>'male'),
    array('id'=>4,'name'=>'Diaochan','score'=>82 ,'sex'=>'female'),
    array('id'=>5,'name'=>'Machao','score'=>76 ,'sex'=>'male'),
    array('id'=>6,'name'=>'Lvbu','score'=>73 ,'sex'=>'male'),
    array('id'=>7,'name'=>'Caocao','score'=>99 ,'sex'=>'male'),
    array('id'=>8,'name'=>'Sunshangxiang','score'=>79 ,'sex'=>'female'),
    array('id'=>9,'name'=>'Dianwei','score'=>87 ,'sex'=>'male'),
    array('id'=>10,'name'=>'Sunce','score'=>56 ,'sex'=>'male')
);
```

步骤 2　为了便于按照分数降序排列，将考生分数保存到数组 $score 中。

```
$score =array();
foreach( $stu_info as $value){
   $score[] =$value['score'];
}
```

步骤 3　$score 是保存考生分数的一维数组，现将该数组按照分数降序排列，且要保存数组的键，因为不同的考生信息在原数组 $stu_info 中对应着不同的考生信息。代码如下。

```
arsort($score);
```

步骤 4　使用排序后的数组重新构建新数组，新数组的键和值与 $score 数组有关。代码如下。

```
$stu_new =array();
foreach($score as $key=>$value){
   $stu_new[] =$stu_info[$key];
}
unset($stu_info);
```

步骤 5　输出新数组，代码如下。

```
echo "<table align='center' border='1' width='400'>";
echo "<tr>";
echo "<th>ID</th><th>name</th><th>score</th><th>sex</th>";
```

```
echo "</tr>";
foreach($stu_new as $value){
   echo "<tr>";
       foreach($value as $v){
         echo "<td>".$v."</td>";
       }
   echo "</tr>";
}
echo "</table>";
```

上面的全部代码保存为 eg32.php。运行该程序,结果如图 6-5 所示。

ID	name	score	sex
7	Caocao	99	male
3	Zhangfei	89	male
9	Dianwei	87	male
4	Diaochan	82	female
8	Sunshangxiang	79	female
1	Liubei	79	male
5	Machao	76	male
2	Guanyu	75	male
6	Lvbu	73	male
10	Sunce	56	male

图 6-5　考生成绩排序结果

6.7　习　　题

一、填空题

1. 根据数组的维数来分,数组可以分为一维数组、________和多维数组。
2. ________函数用于建立一个包含指定范围元素的数组。
3. ________函数用于在数组尾部添加数组元素。
4. ________函数用于在数组头部添加数组元素。
5. ________函数用于从数组头部删除元素。
6. ________函数用于在数组尾部删除元素。
7. ________函数用于对数组进行排序,且保留数组的键。
8. 仅使用数字作为键的数组称为________数组。
9. 使用字符串作为数组的键的数组称为________数组。
10. ________函数对数组进行反向排序。

二、选择题

1. 在删除数组元素时，________函数可以删除数组的第一个元素。

A. array_shift() B. array_pop() C. arrsy_spice() D. unset()

2. 执行下面的代码，程序输入结果为________。

```
<?php
  $a[] = 'apple';
  $a['banana'] = 'banana';
  $a[3] = 'orange';
  $a[] = 'manguo';
  echo $a[2];
?>
```

A. banana B. manguo C. apple D. 什么都不输出

3. PHP 中数组实现排序时，程序员通过________函数来自定义比较函数，从而达到对数组中的值进行排序的目的。

A. ksort() B. usort() C. asort() D. rsort()

4. 执行下面的代码，最终输出结果为________。

```
<?php
  $a =array('a', 'b', 'c', 'd', 'e');
  $b =array_slice($a, 2, -1);
  for( $i=0 ; $i<count($b) ; $i++)
    echo $b[$i];
?>
```

A. c B. cd C. cde D. bcde

5. 程序中要删除数组中出现的重复的内容，可以利用________函数进行清理。

A. array_merge() B. array_flip

C. array_unique() D. array_push()

6. 程序员想从数组中随机抽取若干元素，可以使用________数组。

A. arry_rand() B. array_push()

C. array_splice() D. array_combine()

三、程序设计题

1. 定义一个一维数组，并对它进行排序和输出；
2. 定义一个一维数组，删除其中重复的元素；
3. 定义一个一维数组，使用 array_push()函数向数组中添加一个元素。
4. 定义一个一维数组，使用 array_unshift()函数向数组中添加 3 个元素。
5. 已知保存考生成绩的二维数组如下，输出最高分考生信息。

```
$score =array(
    array('01','liubei',87),
    array('02','guanyu',78),
    array('03','zhangfei',84),
    array('04','zhaoyun',73),
    array('05','machao' ,82),
```

```
        array('06','huangzhong' ,81),
        array('07','weiyan',65),
        array('08','jiangwei',64)
    );
```

6. 将上面的二维数组排序,按照分数高低排序。
7. 随机输出上述二维数组中的 3 个考生的信息。

第 7 章　字符串和正则表达式

知识点：

- 字符串的定义
- 字符串的连接
- 字符串的基础操作
- 字符串编码
- 字符串编码转换
- 字符串加密和解密
- 正则表达式
- 正则表达式的书写
- 正则表达式的使用

本章导读：

字符串是程序设计中非常重要的内容，因为信息的传播和处理很多时候都需要通过字符串来进行。在 PHP 中字符串是非常重要的数据类型，可以把字符串理解为字符型数组。本章将详细讲述字符串的各种操作。

正则表达式是对字符串的限制，正则表达式以表达式的形式限制了字符的格式，验证字符串是否合乎要求。

7.1　字符串概述

在对字符串进行操作之前，必须保证已存在一个字符串，在 PHP 中可以使用三种方法来定义字符串。本节将详细讲解字符串的基础知识及其定义。

7.1.1　字符串基础

文字是信息记录和传播的载体。在 PHP 中字符串又可以理解为字符型数组。一个字符串可以包含若干个字符，这些字符可以是数字、大小写英文字母，以及像%、^、*、$、@等特殊字符，也可以包含汉字。每一个字符占一个字节，汉字所占字节数与编码有关。

【示例 1】 将字符串理解为数组。

eg1.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $st1 = "我爱 PHP";
```

```
  echo $st1[0];
  echo $st1[1];
  echo $st1[2];
  echo $st1[3];
  echo $st1[4];
  echo $st1[5];
  echo $st1[6];
  //用 utf-8 编码,一个汉字占 3 字节,输出为"我爱 P"
  //用 gb2312 编码,一个汉字占 2 字节。输出内容读者可以自己试试
?>
```

7.1.2 字符串连接运算

在 PHP 中可以使用点号“.”连接两个或多个字符串。

【示例 2】 字符串的连接。

eg2.php 代码如下。

```
<?php
  $st1 ="I love ".'PHP';                //I love PHP
  $st1 .=",I love JavaScript too.";     //I love PHP,I love JavaScript too.
  echo $st1;
?>
```

7.1.3 使用定界符定义字符串

在前面的数据类型章节中已经讲述过使用单引号和双引号定义字符串,实际上在 PHP 中还可以使用定界符来定义字符串。

使用定界符定义字符串的方法是:在字符串的前面加符号“<<<”和标识符,在字符串的结束位置另起一行且在最左边使用相同的标识符结尾。使用定界符表示字符串需注意以下几点。

- 字符串不需要使用引号来引用。
- 字符串前后的标识符必须一致,且结尾标识符必须书写在最左边。
- 字符串中可以一些类似于 HTML 的格式,特殊字符也不需要转义。
- 字符串中可以出现变量。

【示例 3】 使用定界符定义字符串。

eg3.php 代码如图 7-1 所示。

```
1  <?php
2    header("content-type:text/html;charset=utf-8");
3    $title="三国人物";
4    $content="刘备、诸葛亮、曹操、孙权、关羽、张飞";
5    $st1 =<<<sanguo
6    <p align="center">$title</p>
7    <font size='-1'>滚滚长江东逝水,浪花淘尽英雄</font>
8    <h4>$content</h4>
9  sanguo;
10   //第9行后面不要跟任何内容,哪怕是注释
11   //第9行标识符必须与第5行标识符sanguo完全相同,
12   //第9行必须顶格书写,不能留空格
13   echo $st1;
14 ?>
```

图 7-1 使用定界符定义字符串

程序运行结果如图 7-2 所示。



图 7-2　示例 3 的程序运行效果

说明：

- 使用定界符定义的字符串中可以使用变量，本程序中就定义了变量 $title 和 $content，还使用了 HTML 标识。对于要实现一定格式的字符串来说，使用定界符来定义字符串比较合适。
- 定界符的起始和终止标识必须完全一致，包括大小写，且终止标识符必须顶格书写，且不能在后面加包括注释在内的任何字符。

7.2　字符串操作

在 PHP 中，与字符串相关的操作很多，字符串操作是各种不同计算机语言中不可或缺的内容，字符串的操作一般都是通过系统定义的字符串函数来进行的。

7.2.1　统计字符串

对字符串的统计包括统计字符串长度、统计字符串中单词个数等操作。与统计字符串有关的一些函数如表 7-1 所示。

表 7-1　字符串统计函数

函 数 名 称	说　　明
str_word_count()	统计字符串中单词的个数
strcspn()	返回在找到任何指定字符之前在字符串中查找的字符数
strlen()	返回字符串长度
count_chars()	返回字符串所用字符的信息

1. 统计字符串长度

在 PHP 中统计字符串的长度是最常见的操作之一，比如控制输入的密码长度必须在某个范围内。PHP 中提供了一个名为 strlen()的函数，用于统计字符串的长度，该函数的语法格式为：

```
int strlen ( string $string )
```

返回给定的字符串 string 的长度。参数 string 是需要计算长度的字符串。

返回值：成功则返回字符串 string 的长度；如果 string 为空，则返回 0。汉字的长度与编码有关。

【示例 4】 统计字符串的长度。

步骤 1 eg4-1.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $st1 ="我爱 PHP";
  echo strlen($st1);                //输出 9,汉字长度为 3
?>
```

步骤 2 eg4-2.php 代码如下。

```
<?php
  header("content-type:text/html;charset=gb2312");
  $st1 ="我爱 PHP";
  echo strlen($st1);                //输出 7,汉字长度为 2
?>
```

2. 获取字符信息

获取字符串中的字符信息，需要使用名为 count_chars()的函数。该函数的语法格式为：

```
mixed count_chars ( string $string [, int $mode =0 ] )
```

统计 string 中每个字节值(0..255)出现的次数，使用多种模式返回结果。参数 string 是需要统计的字符串。mode 参见返回的值。

返回值：根据不同的 mode，count_chars()返回下列不同的结果。

- 0　以每个字节值作为键名，出现次数作为值的数组。
- 1　与返回值 0 相同，但只列出出现次数大于零的字节值。
- 2　与返回值 0 相同，但只列出出现次数等于零的字节值。
- 3　返回由所有使用了的字节值组成的字符串。
- 4　返回由所有未使用的字节值组成的字符串。

【示例 5】 统计字符信息。

步骤 1 eg5-1.php 代码如下。

```
<?php
  $data ="Hello,PHP" ;
  $array =count_chars($data,1);//列出出现次数大于零的字节值,数组$array 键名为字节的
                               ASCII 值
  //$array=count_chars($data,0)统计 ASCII 码为 0~ 255 的全部字节出现的次数,0 次也列出
  print_r($array);
?>
```

程序运行结果如下。

```
array
(
    [44] =>1
```

```
    [72] =>2
    [80] =>2
    [101] =>1
    [108] =>2
    [111] =>1
)
```

说明：

- 返回结果为一个数组，该数组保存着各个字符在字符串中的个数。
- 数组表示的含义是：[44] => 1 表示“,”出现 1 次，[72] => 2 表示“H”出现 2 次，[80] => 2 表示“P”出现 2 次，[101] => 1 表示“e”出现 1 次，[108] => 2 表示“l”出现 1 次，[111] => 1 表示“o”出现 1 次。

步骤 2　eg5-2. php 代码如下。

```
<?php
  $data ="Hello,PHP";
  $str =count_chars($data,3);
  var_dump($str);
?>
```

3. 获取单词信息

使用 str_word_count()函数可以获得一个字符串中单词的信息，该函数的语法格式为：

```
mixed str_word_count ( string $string [, int $format =0 [, string $charlist ]] )
```

该函数用于统计 string 中单词的数量。如果可选的参数 format 没有被指定，那么返回值是一个代表单词数量的整型数。如果指定了 format 参数，返回值将是一个数组，数组的内容则取决于 format 参数。format 的可能值和相应的输出结果如下所列。

有关参数说明如下。

- string：这是一个字符串。format 是指定函数的返回值。当前支持的值如下。
  - ✍0　返回单词的数量，为默认值。
  - ✍1　返回一个包含 string 中全部单词的数组。
  - ✍2　返回关联数组。数组的键是单词在 string 中出现的数值位置，数组的值是这个单词。
- charlist：附加的字符串列表，其中的字符将被视为单词的一部分。

返回值：返回一个数组或整型数，这取决于 format 参数的选择。

【示例 6】　统计单词信息。

eg6. php 代码如下。

```
<?php
  $str ="Hello fri3nd,I'm Ja6ck!";
  print_r(str_word_count($str,1));
  print_r(str_word_count($str,2));
  print_r(str_word_count($str,1,'63'));
  echo str_word_count($str);
?>
```

程序运行结果如图 7-3 所示。

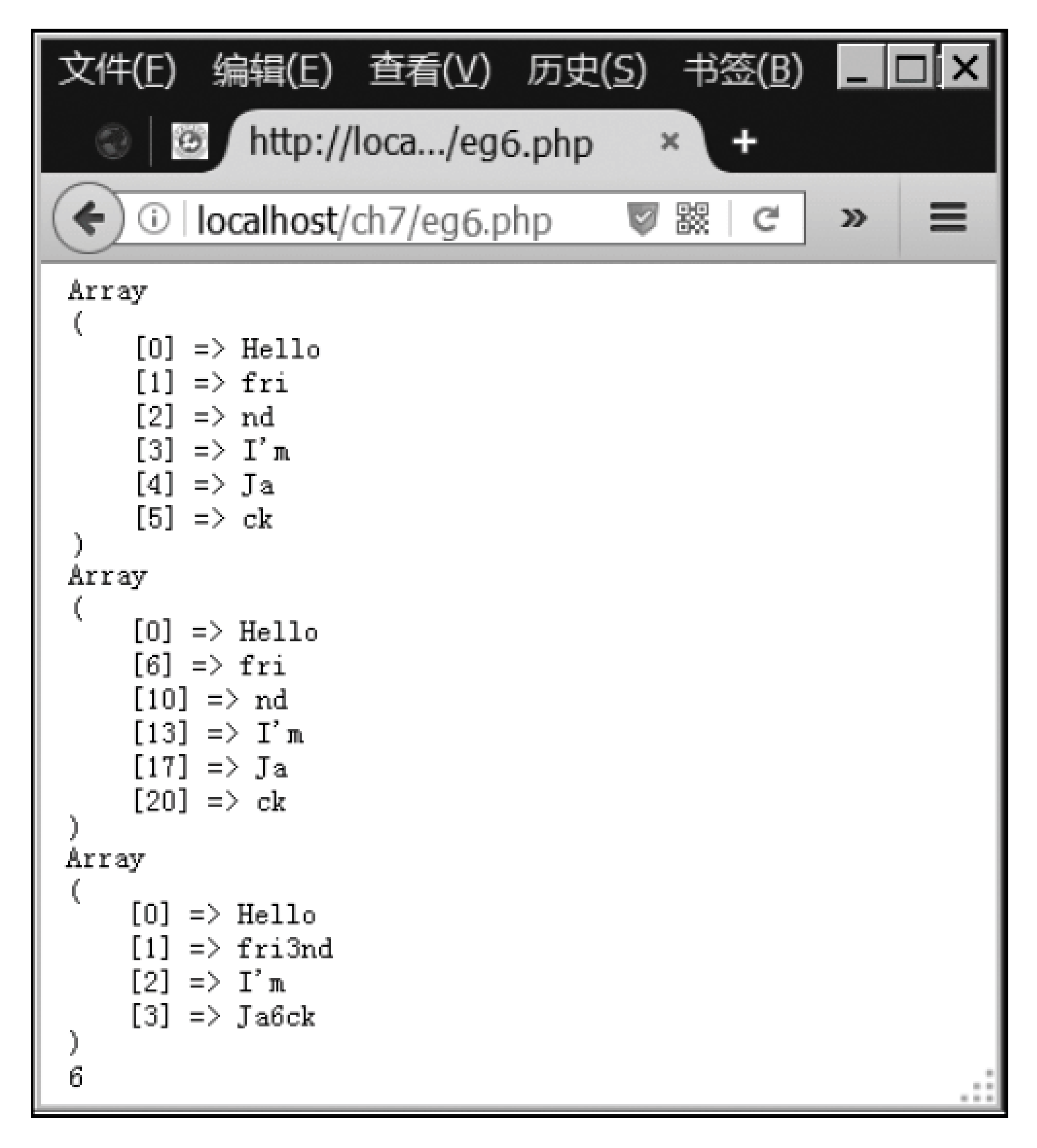


图 7-3　统计单词信息

说明：

- str_word_count($str,1)返回单词数组，键为数字；str_word_count($str,2)返回单词数组，键为单词出现在字符串中的位置；str_word_count($str,1,'63')把字符'6'和'3'看成单词的一部分。
- str_word_count($str)返回单词的个数，不把 charlist 中的字符看成单词的一部分。

4. 使用 strcspn()函数获取不匹配遮罩的起始子字符串的长度

在 PHP 中可以使用名为 strcspn()的函数获取不匹配遮罩的起始子字符串的长度。该函数的语法格式为：

```
int strcspn ( string $str1, string $str2 [, int $start [, int $length ]] )
```

该函数返回值为在 str1 中所有字符都不存在于 str2 范围的起始子字符串的长度。

有关参数说明如下。

- str1：第一个字符串。
- str2：第二个字符串。
- start：查找的起始位置。
- length：查找的长度。

返回值：以整型数返回子串的长度。

【示例 7】 使用 strcspn()函数获取不匹配遮罩的起始子字符串的长度。

eg7. php 代码如图 7-4 所示。

说明：

- strcspn()函数返回 str1 中所有字符都不存在于 str2 范围的起始子字符串的长度，$a、$b、$c 和 $d 对应的范围是整个字符串。

```
<?php
  $a = strcspn('abcd','frc');
  echo $a.'<br />'; //出现'c'之前有2个字符:'a'和'b'
  $b = strcspn('abcd','banana');
  echo $b.'<br />';//出现'a'之前有0个字符
  $c = strcspn('hello','l');
  echo $c.'<br />';//出现'l'字符之前有2个字符:'h'和'e'
  $d = strcspn('hello','word');
  echo $d.'<br />';//出现'o'之前有4个字符:'h','e','l'和'l'
  $e = strcspn('abcdabcdabcd','b',2,5);
  echo $e.'<br />';//蓝色选中部分中'b'之前有3个字符
  $f = strcspn('axbcdabcdabxcd','x',3);
  echo $f.'<br />';//蓝色选中部分中'x'之前有8个字符
?>
```

图 7-4　使用 strcspn()函数获取不匹配遮罩的起始子字符串的长度

- $e 对应的范围是从 2 开始长度为 5 的区域，$f 对应的范围是从 3 开始到最后的区域。

7.2.2　空格和特殊字符

在 PHP 中有一些函数专门用来处理字符串中的空格和特殊字符，如删除字符前后的空格、删除字符的 HTML 标记、删除字符中的反斜线等，其相关的函数如表 7-2 所示。

表 7-2　空格和特殊字符处理函数

函数名称	说　　明
addcslashes()	在指定的字符前添加反斜线
addslashes()	在指定的预定义字符前添加反斜线
strip_tags()	剥去 HTML、XML 和 PHP 的标记
count_chars()	返回字符串所用字符的信息
stripcslashes()	删除由 addcslashes()函数添加的反斜线
stripslashes()	删除由 addslashes()添加的反斜线
quotemeta()	在字符串某些预定义的字符前添加反斜线
rtirm()	从字符串末端开始删除空白字符或者其他预定义字符
trim()	从字符串两端开始删除空白字符或者其他预定义字符
ltrim()	从字符串左边开始删除空白字符或者其他预定义字符
chop()	这是 rtrim()的别名

1. 去掉字符串左侧的指定字符

去掉字符串左侧的指定字符可使用 ltrim()函数，该函数的语法格式为：

```
string ltrim ( string $str [, string $charlist ] )
```

该函数的功能是删除字符串开头的空白字符(或其他字符)，并返回剩余部分。

有关参数说明如下。

- str：输入的字符串。
- charlist：通过 charlist 参数，也可以指定想要删除的字符，简单地列出要删除的所有字符即可。使用 charlist 可以指定字符的范围。

返回值：该函数返回一个删除了 str 最左边的空白字符的字符串。如果不使用第二个

参数，ltrim()仅删除以下字符。

- " "（ASCII 32（0x20）） 普通空白字符。
- "\t"（ASCII 9（0x09）） 制表符。
- "\n"（ASCII 10（0x0A）） 换行符。
- "\r"（ASCII 13（0x0D）） 回车符。
- "\0"（ASCII 0（0x00）） 空字节符。
- "\x0B"（ASCII 11（0x0B）） 垂直制表符。

【示例 8】 使用 ltrim()函数删除字符串左侧的空白字符和预定义字符。

eg8.php 代码如下。

```
<?php
  echo "<pre>";
  $text = "\t\tThese are a few words :) ...";
  echo $text."\n";
  echo ltrim($text)."\n";
  //无第 2 个参数,仅删除回车、换行、制表、垂直制表、空格、空字符
  $text = "a\tbI love my country.";
  echo ltrim($text,"b\ta")."\n";
  //删除\t 控制符,删除字符'a'
  $hello = "Hello World";
  echo ltrim($hello,"Hl");
  //删除'H'而不删除'l',因'l'前面有'e'不在第 2 个参数之列
?>
```

2. 去掉字符串右侧的指定字符

去掉字符串右侧的指定字符可使用 rtrim()函数，该函数的语法格式为：

```
string rtrim ( string $str [, string $charlist ] )
```

该函数的功能是删除 str 末端的空白字符并返回剩余部分。

有关参数说明如下。

- str：输入的字符串。
- charlist：通过 charlist 参数，也可以指定想要删除的字符，简单地列出要删除的所有字符即可。使用 charlist 可以指定字符的范围。

返回值：该函数返回一个删除了 str 最右边的空白字符的字符串。如果不使用第二个参数，rtrim()仅删除以下字符。

- " "（ASCII 32（0x20）） 普通空白符。
- "\t"（ASCII 9（0x09）） 制表符。
- "\n"（ASCII 10（0x0A）） 换行符。
- "\r"（ASCII 13（0x0D）） 回车符。
- "\0"（ASCII 0（0x00）） 空字节符。
- "\x0B"（ASCII 11（0x0B）） 垂直制表符。

【示例 9】 使用 rtrim()函数删除字符串右侧的空白字符和预定义字符。

eg9.php 代码如下。

```
<?php
  echo "<pre>";
  $text = "Hello,world  \t";
  echo strlen($text)."\n";
  echo strlen(rtrim($text))."\n";
  //去掉了两个空格和一个制表符
  $text = "Hello,worldabcd";
  echo rtrim($text,"bd");
  //去掉了'd'而无法去掉'b',因为'b'后还有'c'并不在第 2 个参数中
?>
```

3. 去掉字符串两端的指定字符

去掉字符串两端的空白字符或者指定字符使用 trim()函数，该函数的语法如下。

```
string trim ( string $str [, string $charlist = " \t\n\r\0\x0B" ] )
```

该函数的功能是返回字符串 str 去除首尾空白字符后的结果。如果不指定第二个参数，trim() 将去除以下字符。

- " " (ASCII 32 (0x20))　普通空格符。
- "\t" (ASCII 9 (0x09))　制表符。
- "\n" (ASCII 10 (0x0A))　换行符。
- "\r" (ASCII 13 (0x0D))　回车符。
- "\0" (ASCII 0 (0x00))　空字节符。
- "\x0B" (ASCII 11 (0x0B))　垂直制表符。

有关参数说明如下。

- str：待处理的字符串 。
- charlist：可选参数，过滤字符也可由该参数指定。一般要列出所有希望过滤的字符，也可以使用 charlist 列出一个字符范围。

返回值：返回值为过滤后的字符串。

【示例 10】　使用 trim()函数删除字符串两端的空白字符和预定义字符。

eg10. php 代码如下。

```
<?php
  echo "<pre>";
  $text = "  \tHello,world  \t";
  echo strlen($text)."\n";
  echo strlen(trim($text))."\n";
  //去掉两边的空格、制表符
  $text = "dbabcHello,worldabcdd";
  echo trim($text,"bd");
  //去掉左边的'd'和'b',右边的两个'd'
?>
```

7.2.3　大小写转换

大小写转换函数的使用，可以有效地规范字符串的输出格式，如验证码不区分大小写

时，要将用户输入的字符大小写统一，再如将字符串的首字母转换为大写，等等。PHP 中大小写转换的函数如表 7-3 所示。

表 7-3　大小写转换函数

函 数 名 称	说　明
strtolower()	字符串变成小写
strtoupper()	字符串变成大写
ucfirst()	首字母变大写
ucwords()	字符串中每个单词首字母变成大写

1. 将字符串中的大写变小写

将字符串中的大写变小写可使用 strtolower()函数，该函数的语法格式为：

```
string strtolower ( string $string )
```

该函数的功能是将 string 中所有的字母字符转换为小写。

说明：string 表示输入字符串。

返回值：返回转换后的字符串。

【示例 11】 使用 strtolower()函数将字符串变成小写。

eg11.php 代码如下。

```
<?php
  $st1 ="I'm a teacher,my telephone is 138 * * * 8888.";
  $st2 =strtolower($st1);           //将大写英文字符变成小写英文字符,其他字符不变
  echo $st2;
?>
```

2. 将字符串中的小写变大写

将字符串中的大写变小写可以使用函数 strtoupper()，该函数的语法格式为：

```
string strtoupper ( string $string )
```

该函数的功能是将 string 中所有的英文字母字符转换为大写字符，其他字符保持不变。

说明：string 表示输入字符串。

返回值：返回转换后的字符串。

【示例 12】 使用 strtoupper()函数将字符串变成小写。

eg12.php 代码如下。

```
<?php
  $st1 ="I'm a teacher,my telephone is 138 * * * 8888.";
  $st2 =strtoupper($st1);  //将小写英文字符变成大写英文字符,其他字符不变
  echo $st2;
?>
```

3. 查找字符串的首次出现

查找字符串首次出现的位置可使用 strpos()函数，该函数的语法格式为：

```
mixed strpos ( string $haystack, mixed $needle [, int $offset =0 ] )
```

返回 needle 在 haystack 中首次出现的数字位置。

有关参数说明如下。

- haystack：表示要在该字符串中进行查找。
- needle：如果 needle 不是一个字符串，那么它将被转换为整型并被视为字符的顺序值。
- offset：如果提供了此参数，搜索会从字符串中该字符数的起始位置开始统计。

返回值：返回 needle 存在于 haystack 字符串起始的位置(独立于 offset)。同时注意字符串位置是从 0 开始，而不是从 1 开始的。如果没找到 needle，将返回 false。

【示例 13】 使用 strpos()函数查找字符串位置。

eg13.php 代码如下。

```
<?php
  echo "<pre>";
  $mystring='I like Wuhan,Wuhan is a beautiful city.' ;
  $findme='Wuhan' ;
  $pos1=strpos ( $mystring,  $findme );  //从位置 0 开始查找
  if ( $pos1  ===  false ) {              //必须是===,不能用==
    echo  "The string '$findme' was not found in the string '$mystring'" ;
  } else {
    echo  "The string '$findme' was found in the string '$mystring'" ;
    echo  " and exists at position  $pos1.\n" ;
  }
  $pos2=strpos ( $mystring, $findme, 8);  //查找位置是从 8 开始的,并向后查找
  if ( $pos2===false ) {
    echo  "The string '$findme' was not found in the string '$mystring'" ;
  } else {
    echo  "The string '$findme' was found in the string '$mystring'" ;
    echo  " and exists at position  $pos2." ;
  }
?>
```

说明：此处用了“===”判断是否为子串，因为 0==false 是成立的。请看示例 14。

【示例 14】 使用 strpos()函数查找字符串位置。

eg14.php 代码如下。

```
<?php
  $mystring = 'Wuhan is a beautiful city,I like Wuhan.' ;
  $findme = 'Wuhan' ;
  $pos = strpos ( $mystring, $findme );
  if ( $pos == false ) {
    echo "The string '$findme' was not found in the string '$mystring'" ;
  } else {
    echo "The string '$findme' was found in the string '$mystring'" ;
    echo " and exists at position $pos." ;
  }
?>
```

说明：本例得出的错误结论是——'Wuhan'不在字符串'Wuhan is a beautiful city,I

like Wuhan. '中。错误原因是 strpos ($ mystring, $ findme)的值是 0,而 0==false 是成立的。因此判断一个字符串是否是另一个字符串的子串时,一定要使用"==="或"!=="。

默认查找位置是从 0 开始的,可以指定查找的初始位置。

4. 将字符串的首字母转换为大写

将字符串的首字母转换为大写可使用 ucfirst()函数,该函数的语法格式为:

```
string ucfirst ( string $ str )
```

将 str 的首字符(如果首字符是字母)转换为大写字母,并返回这个字符串。

说明:$ str 参数表示要转换的字符串。

返回值:返回结果字符串。

【示例 15】 使用 ucfirst()函数将首字母转换为大写。

eg15. php 代码如下。

```
<?php
  echo "<pre>";
  $ foo = 'hello world!' ;
  $ foo =ucfirst ( $ foo );                //Hello world!
  echo $ foo."\n";
?>
```

5. 将字符串的每个单词的首字符(如果首字符是英文字母)转换为大写字母

将字符串每个单词的首字符转换为大写字母可使用 ucwords()函数,该函数的语法格式为:

```
string ucwords ( string $ str )
```

将 str 中每个单词的首字符(如果首字符是字母)转换为大写字母,并返回这个字符串。

这里单词的定义是紧跟在空白字符(空格符、制表符、换行符、回车符、水平线以及竖线)之后的子字符串。

说明:$ str 参数表示输入字符串。

返回值:返回转换后的字符串。

【示例 16】 使用 ucwords()函数将单词的首字母转换为大写。

eg16. php 代码如下。

```
<?php
  echo "<pre>";
  $ foo = 'hello world!' ;
  $ foo =ucwords  ( $ foo );              //Hello World!
  echo $ foo."\n";
?>
```

7.2.4 分隔字符串

PHP 提供了多种分割字符串方法,如将字符串分隔为一连串更小的字符串,或者把字符串打散为数组等。有关字符串的函数如表 7-4 所示。

表 7-4　分隔字符串的函数

函 数 名 称	说　　明
chunk_split()	把字符串分隔为一连串更小的部分
explode()	把字符串打散为数组
implode()和 join()	把数组组合为字符串
str_split()	把字符串分隔到数组中
strtok()	把字符串分隔为更小的字符串

1. strtok()函数

标记分隔字符串函数 strtok(),该函数用于将字符串 str 分隔为若干子字符串。语法格式为:

```
string strtok ( string $str, string $token )
string strtok ( string $token )
```

strtok()函数将字符串 $str 分隔为若干子字符串,每个子字符串以 token 中的字符分隔。这也就意味着,如果有个字符串是 "This is an example string",则可以使用空格字符将这句话分隔成独立的单词。

注意仅第一次调用 strtok()函数时使用 string 参数。后来每次调用 strtok,都将只使用 token 参数,因为它会记住它在 string 字符串中的位置。如果要重新开始分隔一个新的字符串,需要再次使用 string 来调用 strtok()函数,以便完成初始化工作。注意可以在 token 参数中使用多个字符。字符串将被该参数中任何一个字符分隔。

有关参数说明如下。

- str:表示要分隔的字符串。
- token:分隔 $str 时使用的分界字符。

返回值:返回标记后的字符串。

【示例 17】 使用 strtok()函数分隔字符串。

eg17.php 代码如下。

```
<?php
  echo "<pre>";
  $str = "Wuhan,Nanjing Beijing,Shanghai * Changsha";
  $result = strtok($str,", *  ");   //第一次调用使用两个参数
  while($result!=false){
    echo $result."\n";
    $result = strtok(", *  ");      //后面的调用只使用一个参数
  }
?>
```

程序运行结果如图 7-5 所示。

说明:

- 第一次调用 strtok()函数时使用 string 参数,后面的调用仅使用第二个参数,函数会记住 token 在字符串 string 中的位置。
- 本例 strtok()函数中包含 3 个字符,其中一个是空格。



图 7-5　strtok()函数的程序运行结果

- 当 string 中不包含 token 字符时，返回 false，可以用于结束循环控制。

2. explode()函数

使用一个字符串分隔另一个字符串为数组时使用 explode()函数。该函数的语法格式为：

```
array explode ( string $delimiter, string $string [, int $limit ] )
```

此函数返回由字符串组成的数组，每个元素都是 string 的一个子串，它们被字符串 delimiter 作为边界点分隔出来。

有关参数说明如下。

- $delimiter：边界上的分隔字符。
- $string：输入的字符串。
- $limit：如果设置了 $limit 参数并且是正数，则返回的数组最多包含 $limit 个元素，而最后那个元素将包含 $string 的剩余部分；如果 $limit 参数是负数，则返回除了最后的 $limit 的绝对值个元素外的所有元素；如果 $limit 是 0，则会被当作 1。

返回值：此函数返回由字符串组成的 array，每个元素都是 string 的一个子串，它们被字符串 $delimiter 作为边界点分隔出来。如果 $delimiter 为空字符串("")，explode()将返回 false。如果 $delimiter 所包含的值在 $string 中找不到，并且使用了负数的 $limit，那么会返回空的 array，否则返回包含 $string 单个元素的数组。

【示例 18】 使用 explode()函数分隔字符串。

eg18.php 代码如下。

```
<?php
  echo "<pre>";
  $input1 = "hello" ;
  $input2 = "hello,there" ;
  var_dump(explode(',',$input1));     //数组含有一个元素，即 hello
  var_dump(explode(',',$input2));     //数组含有两个元素，即 hello、there

  $str = 'one|two|three|four';
  //正数的 limit
  print_r(explode('|',$str,2));  //分隔为两部分，而最后那个元素将包含 string 的剩余部分
  //负数的 limit(自 PHP 5.1 起)
  print_r(explode('|',$str,-1));     //返回除最后一个(即|-1|)元素外的全部元素
?>
```

程序运行结果如图 7-6 所示。

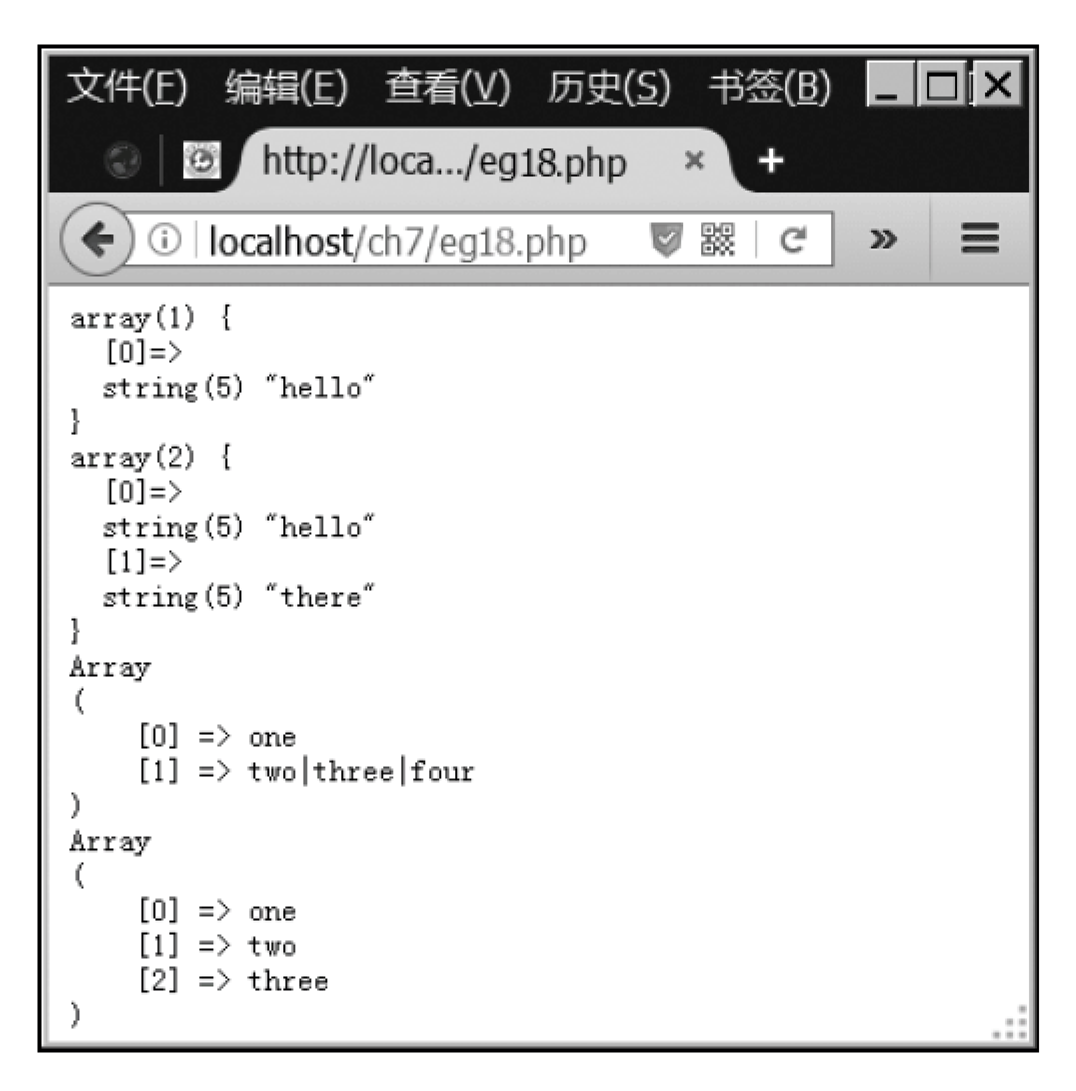


图 7-6　应用 explode()函数的程序运行结果

3. implode()函数

将一个一维数组的值转化为字符串可使用 implode()函数。该函数的语法格式为：

```
string implode ( string $glue, array $pieces )
string implode ( array $pieces )
```

该函数的功能是用 $ glue 将一维数组的值连接为一个字符串。

有关参数说明如下。

- $ glue：默认为空的字符串。
- $ pieces：要转化的数组。

返回值：返回一个字符串，其内容为由 glue 分隔开的数组的值。

【示例 19】 使用 implode()函数将数组连接成字符串。

eg19.php 代码如下。

```
<?php
  echo "<pre>";
  $array1 = array('lastname','email','phone');
  $str1 = implode(",",$array1);          //使用逗号连接各个数组元素
  var_dump($str1);                       //string(20),即 "lastname,email,phone"
  $st2 = implode('hello',array());       //空数组返回空字符串
  var_dump($st2);                        //string(0),即 ""
  $array2 = array("Wuhan","Changsha","Beijing");
  $str3 = implode($array2);              //使用空字符连接
  var_dump($str3);                       //string(20),即 "WuhanChangshaBeijing"
?>
```

7.2.5　截取字符串

字符串的截取与字符串的分隔不一样，字符串的分隔是将长的字符串分散为多个小的

字符串;而字符串的截取是将字符串的一部分提取出来。例如,要在一个包含职工信息的字符串中截取职工的姓名信息,就属于字符串的截取。字符串的截取函数以及有关说明如表 7-5 所示。

表 7-5 字符串的截取函数

函数名称	说 明
strstr()	查找字符串首次出现的位置,并返回第一次出现的位置到结尾的字符串
stristr()	与 strstr()函数类似,不同之处在于前者在查找时区分大小写,后者在查找时不区分大小写
strpos()	查找字符串第一次出现的位置
strrchr()	字符串最后一次出现到最后的部分
substr()	返回字符串的一部分

1. strstr()函数

在 PHP 中查找字符串的首次出现,可以使用 strstr()函数。该函数的语法格式为:

```
string strstr ( string $haystack, mixed $needle [, bool $before_needle = false ] )
```

该函数返回 $ haystack 字符串从 $ needle 第一次出现的位置开始到 $ haystack 结尾的字符串。

该函数区分大小写。如果想要不区分大小写,请使用 stristr()函数 。如果仅需要确定位置,则 strpos()函数速度更快。

有关参数说明如下。

- $ haystack:输入的字符串。
- $ needle:如果 $ needle 不是一个字符串,那么它将被转化为整型并且作为字符的序号来使用。
- $ before_needle:若为 true,则返回 needle 在 haystack 中的位置之前的部分。

返回值:返回字符串的一部分或者 false(如果未发现 needle)。

【示例 20】 使用 strstr()函数查找子串。

eg20. php 代码如下。

```
<?php
  echo "<pre>";
  $email    =   'name@example.com' ;
  $domain = strstr ( $email, '@' );        //若有多个@符号,则只考虑第一个@符号
  echo   $domain."\n" ;                    //打印 @example.com
  $user = strstr ($email, '@', true );     //true 表示返回查询字符之前的部分
  echo   $user."\n" ;                      //打印 name
  $st = "wuhanchangsha";
  $st1 = strstr($st, 'ka');                //不存在 ka
  $st2 = strstr($st, 'ka',true);
  var_dump($st1,$st2);                        //因找不到要查找的字符串,$st1 和$st2 都是 false
?>
```

stristr()函数与 strstr()函数类似,不同之处仅在于是否区分大小写,这里不再赘述,读者可以自己查看相关手册。

2. strpos()函数

在 PHP 中，strpos()函数用于查找子字符串在字符串中出现的位置，并返回该位置。该函数的语法格式为：

```
mixed strpos ( string $haystack, mixed $needle [, int $offset = 0 ] )
```

该函数查找子字符串 $needle 在字符串 $haystack 中出现的位置。

有关参数说明如下。

- $haystack：在该字符串中进行查找。
- $needle：要查找的子字符串。
- $offset：从什么位置开始查找，默认位置是第一个字符，即 $offset=0。

返回值：返回 $needle 存在于 $haystack 字符串起始的位置（独立于 $offset）。同时注意字符串位置是从 0 开始，而不是从 1 开始的。如果没找到 $needle，将返回 false。

【示例 21】 使用 strpos()函数求子字符串的位置。

eg21.php 代码如下。

```
<?php
  $mystring = 'abc';
  $findme = 'a';                      //写成 97 也可以，因为 97 是 a 的 ASCII 码
  $pos = strpos($mystring,$findme);
  if ($pos === false) {      //"==="对类型和值都有要求，0== false 成立，而 0=== false 则不
                               成立
    echo  "The string '$findme' was not found in the string '$mystring'";
  } else {
    echo  "The string '$findme' was found in the string '$mystring'";
    echo  " and exists at position $pos";
  }
  echo "<br />";
  if ($pos !== false ) {                     //不能使用“!=”，理由和“===”类似
     echo "The string '$findme' was found in the string '$mystring'";
     echo " and exists at position $pos" ;
  }else{
     echo "The string '$findme' was not found in the string '$mystring'";
  }
  echo "<br />";
  $newstring = 'abcdef abcdef';
  $pos = strpos ($newstring,'a',1);      //从第 2 个位置开始查询，因此$pos = 7，不是 0
  echo $pos;
?>
```

3. strrchr()函数

在 PHP 中，使用 strrchr()函数查找指定字符在字符串中的最后一次出现情况。该函数的语法格式为：

```
string strrchr ( string $haystack, mixed $needle )
```

该函数返回 $haystack 字符串中的一部分，这部分以 $needle 的最后出现位置开始，直到 $haystack 末尾。

有关参数说明如下。

- $ haystack：在该字符串中查找。
- $ needle：如果 $ needle 包含了不止一个字符，那么仅使用第一个字符。该行为不同于 strstr()。如果 $ needle 不是一个字符串，那么将被转化为整型并被视为字符顺序值。

返回值：该函数返回字符串的一部分。如果 $ needle 未被找到，返回 false。

【示例 22】 使用 strrchr()函数查找字符串最后一次出现的位置，并求子串。

eg22.php 代码如下。

```
<?php
  $st1 = 'abadbcefgbcdef';
  $st2 = strrchr($st1,"bc");      //如果有 bc，则找 bc 最后一次出现的位置到最后部分，即 bcdef
  echo $st2;
  echo "<br />";
  $st3 = 'abadbefg';
  $st4 = strrchr($st3,"bc");      //如果无 bc 但有 b，则找 b 最后一次出现的位置到最后部分，
                                    即 befg
  echo $st4;
  echo "<br />";
  $st5 = 'abadbefg';
  $st6 = strrchr($st5,"ka");      //如果既无 ka 也无 k(虽有 a，但不考虑)，则返回 false
  var_dump($st6);
?>
```

4. substr()函数

在 PHP 中，substr()函数用于求子串。该函数的语法格式为：

```
string substr ( string $string, int $start [, int $length ] )
```

该函数返回字符串 $ string 中由 $ start 和 $ length 参数指定的子字符串组成。

有关参数说明如下。

- $ string：输入字符串。
- $ start：如果 $ start 是非负数，返回的字符串将从 $ string 的 $ start 位置开始，并从 0 开始计算。例如，在字符串"abcdef"中，在位置 0 的字符是"a"，位置 2 的字符是"c"等。如果 $ start 是负数，返回的字符串将从 $ string 结尾处向前数第 $ start 个字符开始。如果 $ string 的长度小于或等于 $ start，将返回 false。
- $ length：如果提供了正数的 $ length，返回的字符串将从 $ start 处开始最多包括 $ length 个字符（取决于 $ string 的长度）。如果提供了负数的 $ length，那么 $ string 末尾处的许多字符将会被漏掉（若 $ start 是负数，则从字符串尾部算起）。如果 $ start 不在这段文本中，那么将返回一个空字符串。如果提供了值为 0、false 或 NULL 的 $ length，那么将返回一个空字符串。如果没有提供 $ length，返回的子字符串将从 $ start 位置开始直到字符串结尾。

返回值：返回提取的子字符串，或者在失败时返回 false。

【示例 23】 使用 substr()函数求子串。

eg23.php 代码如下。

```
<?php
  echo '<pre>';
  echo substr('abcdef', 1)."\n";                   //即 bcdef。第 2 个到最后
  echo substr('abcdef', 1, 3 )."\n";               //即 bcd。从第 2 个开始,长度为 3
  echo substr('abcdef', 0, 4 )."\n";               //即 abcd。从第 1 个开始,长度为 4
  echo substr('abcdef', 0, 8 )."\n";               //即 abcdef。从第 1 个开始,长度为 8
  echo substr('abcdef',-4, 2 )."\n";               //即 cd。从-4(即从 c 开始)开始,长度为 2
  echo substr('abcdef',-9 )."\n";                  //即 abcdef。返回最后的 9 个字符
  $string = 'abcdef' ;
  echo $string[3]."\n";                            //即 d
  echo $string[strlen($string) -1]."\n";           //即 f
?>
```

7.2.6　填充字符串或删除字符串

填充字符串是指向给定字符串中某个位置中填充指定个数和指定的字符。例如,向字符串"PHP"后面添加空格,使其长度刚好为 15。再如在首行文本中添加两个空格,在文本最后添加句号等。删除字符串主要有 stripcslashes()函数和 stripslashes()函数,以及前面讲到的一些函数,如 trim()函数等。在 PHP 中有多个用于填充字符串的函数,见表 7-6。

表 7-6　字符串填充函数和删除函数

函数名称	说　明
str_pad()	使用另一个字符串填充字符串为指定长度
str_repeat()	重复一个字符串
nl2br()	在字符串所有新行之前插入 HTML 换行标记
addslashes()	使用反斜线引用字符串
addcslashes()	以 C 语言风格并使用反斜线转义字符串中的字符
stripcslashes()	反引用一个使用 addcslashes()函数转义的字符串
stripslashes()	反引用一个引用字符串

1. str_pad()函数

在 PHP 中,若想使用另一个字符串填充字符串为指定长度,则可以使用 str_pad()函数。该函数的语法格式为:

```
string str_pad ( string $input, int $pad_length [, string $pad_string = " " [, int $pad_type
= STR_PAD_RIGHT ]] )
```

该函数返回 $input 从左端、右端或者同时两端被填充到指定长度后的结果。如果可选的 $pad_string 参数没有被指定,$input 将被空格字符填充,否则它将被 $pad_string 填充到指定长度。

有关参数说明如下。

- $input:输入字符串。
- $pad_length:如果 pad_length 的值是负数,或者小于或者等于输入字符串的长度,不会发生任何填充。
- $pad_string:要填充的字符串默认是空格。如果填充字符的长度不能被 pad_

string 整除,那么 pad_string 可能会被缩短。

- $pad_type：可选的 pad_type 参数的可能值为 STR_PAD_RIGHT、STR_PAD_LEFT 或 STR_PAD_BOTH 。如果没有指定 pad_type,则假定它是 STR_PAD_RIGHT 。

返回值：返回填充后的字符串。

【示例 24】 使用 str_pad()函数填充字符串。

eg24. php 代码如下。

```
<?php
  echo "<pre>";
  $input ="Alien" ;
  echo str_pad($input,10)."\n";
    //输出"Alien     "。默认填充空格
  echo str_pad($input,9,"-=",STR_PAD_LEFT )."\n";
    //只因受长度所限,输出"-=-=Alien",而非"-=-=-Alien"
    //STR_PAD_LEFT 表示在左边填充,"-="是要填充的部分
  echo str_pad($input,10,"_",STR_PAD_BOTH )."\n";
    //输出"__Alien___",两边填充。若无法两边相等,则左边多于右边
  echo str_pad($input,6,"__")."\n";
    //输出"Alien_"。受长度所限,只能填充一个"_"符号
?>
```

2. str_ repeat()函数

在 PHP 中,如果想要重复一个字符串,可以使用 str_ repeat()函数。该函数的语法格式为：

```
string str_repeat ( string $input, int $multiplier )
```

该函数返回 input 重复 multiplier 次后的结果。

有关参数说明如下。

- $input：待操作的字符串。
- $multiplier：$input 被重复的次数。$multiplier 必须大于等于 0。如果 $multiplier 被设置为 0,函数返回空字符串。

返回值：返回重复后的字符串。

【示例 25】 使用 str_ repeat 函数填充字符串。

eg25. php 代码如下。

```
<?php
  echo "<pre>";
  echo str_repeat("-=",10)."\n";             //输出-=-=-=-=-=-=-=-=-=-=
  var_dump(str_repeat("-=",0));              //空字符
?>
```

3. nl2br()函数

在 PHP 中,如果想要在字符串新行之前插入 HTML 换行标记,则可以使用 nl2br()函数,该函数的语法格式为：

```
string nl2br ( string $string [, bool $is_xhtml =true ] )
```

在字符串 string 所有新行之前插入 '
' 或 '
'，并返回。

有关参数说明如下。

- $string：输入的字符串。
- $is_xhtml：确定是否使用 XHTML 兼容换行符。

返回值：返回调整后的字符串。

【示例 26】 使用 nl2br()函数在新行前加 HTML 标记
。

eg26.php 代码如下。

```
<?php
  echo nl2br("foo isn't\n bar");              //输出 foo isn't bar,另起一行输出
  echo nl2br("Welcome\r\nThis is my HTML document",false);
   //输出 This is my HTML document,另起一行输出
  $string = "This\r\nis\n\ra\nstring\r";     //输出 This、is、a、string,都另起一行输出
  echo nl2br($string);
?>
```

4. addslashes()函数

SQL 注入攻击是黑客攻击网站最常用的手段。如果有的站点没有使用严格的用户输入检验，那么常容易遭到 SQL 注入攻击。SQL 注入攻击通常通过给站点数据库提交不良的数据或查询语句来实现，很可能使数据库中的记录遭到暴露、更改或被删除。

为了防止 SQL 注入攻击，PHP 自带一个功能，可以对输入的字符串进行处理，可以在底层对输入进行安全上的初步处理，即 Magic Quotes(php.ini magic_quotes_gpc)。默认情况下开启该选项。如果 magic_quotes_gpc 选项启用，那么输入的字符串中的单引号、双引号和其他一些字符前将会被自动加上反斜线。

但 Magic Quotes 并不是一个很通用的解决方案，没能屏蔽所有有潜在危险的字符，并且在许多服务器上 Magic Quotes 并没有被启用，所以我们还需要使用其他多种方法来防止 SQL 注入。

许多数据库本身就提供这种输入数据处理功能。例如，PHP 的 MySQL 操作函数中有 addslashes()、mysql_real_escape_string()、mysql_escape_string()等函数，可将特殊字符和可能引起数据库操作出错的字符转义。下面讲解 addslashes()函数。

在 PHP 中常常使用 addslashes()函数在字符的前面加反斜线，这样做可以在一定程度上防止 SQL 注入。addslashes()函数的语法格式为：

```
string addslashes ( string $str )
```

该函数返回字符串。该字符串为了数据库查询语句等的需要，在某些字符前加上了反斜线。这些字符是单引号(')、双引号(")、反斜线(\)与 NUL(NULL 字符)。

一个使用 addslashes()函数的例子是当要往数据库中输入数据时，例如，将名字 O'reilly 插入数据库中，这就需要对其进行转义。强烈建议使用 DBMS 指定的转义函数[比如 MySQL 是 mysqli_real_escape_string()，PostgreSQL 是 pg_escape_string()]。但是如果使用的 DBMS 没有一个转义函数，并且使用“\”来转义特殊字符，则可以使用这个函数。仅仅是为了获取插入数据库的数据，额外的“\”并不会插入。当 PHP 指令 magic_quotes_sybase

被设置成 on 时,意味着会进行字符的转义。

PHP 5.4 之前,PHP 指令 magic_quotes_gpc 默认是 on,实际上所有的 GET、POST 和 cookie 数据都被 addslashes()函数加反斜线了。不要对已经被 magic_quotes_gpc 转义过的字符串使用 addslashes(),因为这样会导致双层转义。遇到这种情况时可以使用 get_magic_quotes_gpc()函数进行检测。

说明:$str 参数表示要转义的字符串。

返回值:返回转义后的字符串,也就是加了反斜线的字符串。

【示例 27】 使用 addslashes()函数对字符串进行转义。

eg27.php 代码如下。

```
<?php
  $str1 =addslashes("I'm a teacher. I said:\"You are a good student.\"");
    //在单引号前加反斜线
  echo $str1;
  echo "<br />";
  $str2=addslashes('She said: "I love my country."');
    //在默认字符前添加反斜线。本行在双引号前加反斜线
  echo $str2;
    //单引号、双引号、反斜线、NULL 是默认字符
?>
```

5. addcslashes()函数

在 PHP 中,常常使用 addcslashes()函数在字符的前面加反斜线,这样做可以在一定程度上防止 SQL 注入。该函数类似于 addslashes()函数,但这是类似 C 语言风格的函数。除了默认的字符外,还可以指定要加反斜线的字符列表。addcslashes()函数的语法格式为:

```
string addcslashes ( string $str, string $charlist )
```

有关参数说明如下。

- $str:指要转义的字符。
- $charlist:如果 charlist 中包含有“\n”“\r”等字符,将以 C 语言风格转换,而其他非字母数字且 ASCII 码低于 32 以及高于 126 的字符均转换成使用八进制表示。当定义 charlist 参数中的字符序列时,需要确实知道介于自己设置的开始及结束范围之内的都是一些什么字符。

返回值:返回转义后的字符。

【示例 28】 使用 addcslashes()函数对字符串进行转义。

eg28.php 代码如下。

```
<?php
  $str1 =addcslashes("foo[ ],I'm a teacher.",'a..z');//在小写英文字符前面加反斜线
  echo $str1;
  echo "<br />";
  $str2=addslashes("is your name o'reilly?");        //在默认字符前添加反斜线
  echo $str2;                                      //单引号、双引号、反斜线、NULL 是默认字符
  echo "<br />";
  $str3 =addcslashes("CACBDEGD",'E..A');           //E 的序号大于 A,产生警告信息
```

```
  echo $str3;                          //只将 A 和 E 转义
?>
```

6. stripcslashes()函数

在 PHP 中，常常使用 addcslashes()函数在字符的前面加反斜线。如果要去掉使用 addcslashes()函数所加的反斜线，则要使用 stripcslashes()函数。该函数的语法格式为：

```
string stripcslashes ( string $str )
```

该函数反引用一个使用 addcslashes()函数转义的字符串。

说明：$str 参数代表需要反转义的字符。

返回值：返回反转义后的字符串。

【示例 29】 使用 stripcslashes()函数对字符串进行反转义。

eg29.php 代码如下。

```
<?php
  $str1=addslashes("is your name o'reilly?");
  echo stripcslashes ($str1);
  echo "<br />";
  $str2 =stripcslashes (addcslashes("CACBDEGD",'E..A'));
  echo $str2;
?>
```

7. stripslashes()函数

在 PHP 中，常常使用 addslashes()函数在字符的前面加反斜线。如果要去掉使用 addslashes()函数所加的反斜线，则要使用 stripslashes()函数。该函数的语法格式为：

```
string stripslashes ( string $str )
```

该函数反引用一个使用 addslashes()函数转义的字符串。

说明：$str 参数代表需要反转义的字符。

返回值：返回反转义后的字符串。

【示例 30】 使用 stripslashes()函数对字符串进行反转义。

eg30.php 代码如下。

```
<?php
  $str1 =addslashes("I'm a teacher. I said:\"You are a good student.\"");
  echo stripslashes ($str1);
  echo "<br />";
  $str2=addslashes('She said: "I love my country."');
  echo stripslashes ($str2);
?>
```

7.2.7　比较字符串

字符串比较是指两个字符串之间的对比，可比较两个字符串的长度、子字符串在字符串中出现的次数、两个字符串中匹配字符的数量等，字符串比较函数见表 7-7。

表 7-7　字符串比较函数

函 数 名 称	说　　明
substr_compare()	从指定的偏移位置开始比较两个字符串
substr_count()	计算子串在字符串中出现的次数
strnatcasecmp()	比较两个字符串,不区分大小写
strnatcmp()	比较两个字符串,区分大小写
strncasecmp()	在比较两个字符串时使用长度,不区分大小写
strcmp()	在比较两个字符串时使用长度,区分大小写
similar_text()	计算两个字符串的匹配字符的数量

1. substr_compare()函数

在 PHP 中,substr_compare()函数用于比较字符串开头的若干个字符(不区分大小写)。该函数的语法格式为:

```
int substr_compare ( string $main_str, string $str, int $offset [, int $length
[, bool $case_insensitivity = false ]] )
```

substr_compare()函数从偏移位置 offset 开始比较 main_str 与 str,比较长度为 length 个字符。

有关参数说明如下。

- $main_str:要比较的第一个字符串。
- $str:要比较的第二个字符串。
- $offset:比较开始的位置。如果为负数,则从字符串结尾处开始算起。
- $length:比较的长度。默认值为 str 的长度与 main_str 的长度分别减去位置偏移量 offset 后二者中的较大者。
- $case_insensitivity:如果 case_insensitivity 为 true,比较时将不区分大小写。

返回值:如果 $main_str 从偏移位置 $offset 起的子字符串小于 $str,则返回小于 0 的数;如果大于 $str,则返回大于 0 的数;如果二者相等,则返回 0;如果 $offset 大于等于 $main_str 的长度或 $length 被设置为小于 1 的值,$substr_compare() 将打印出一条警告信息并且返回 false。

【示例 31】 使用 substr_compare()函数比较两个字符串的大小。

eg31.php 代码如下。

```
<?php
  echo "<pre>";
  echo substr_compare("world","or",1,2)."\n";
  //输出 0。从第一个字符串的第二个字符开始比较
  echo substr_compare("world","ld",-2,2)."\n";
  //输出 0。从第一个字符串的倒数第二个字符开始比较,比较长度是 2
  echo substr_compare("world","ork",1,2)."\n";
  //输出 0。从第一个字符串的第二个字符开始比较,比较长度是 2
  echo substr_compare("world","OR",1,2,TRUE)."\n";
  //输出 0。从第一个字符串的第一个字符开始比较,比较长度是 2,不区分大小写
  echo substr_compare("world","or",1,3)."\n";
  //输出 1。从第一个字符串第二个字符开始比较,比较长度为 3
```

```
  echo substr_compare("world","rl",1,2)."\n";
  //输出-1。从第一个字符串第二个位置开始比较,比较长度为 2
  echo substr_compare("Hello world","worldh",6);
  //输出-1。从第一个字符串的第七个字符开始比较,比较长度为全部
  echo substr_compare("abcde","abc",5,1);
  //产生警告,这是因为比较位置超出了字符串长度
?>
```

说明:

- 比较位置是指第一个字符串中的位置。
- 比较长度是两个字符串的长度。
- 第五个参数默认值是 false,表示区分大小写。如果不区分大小写则显示值为 true。

2. substr_count()函数

在 PHP 中,使用 substr_count()函数计算字符串出现的次数。该函数的语法格式为:

```
int substr_count ( string $haystack, string $needle [, int $offset = 0 [, int $length ]] )
```

substr_count()返回子字符串 needle 在字符串 haystack 中出现的次数。注意 needle 区分大小写。该函数不会计算重叠字符。

有关参数说明如下。

- $haystack:在此字符串中进行搜索。
- $needle:要搜索的字符串。
- $offset:开始计数的偏移位置。
- $length:指定偏移位置之后的最大搜索长度。如果偏移量加上这个长度的和大于 haystack 的总长度,则打印警告信息。

返回值:该函数返回整型数,就是出现的子字符的个数。

【示例 32】 使用 substr_count()函数统计子字符出现的次数。

eg32.php 代码如下。

```
<?php
  echo "<pre>";
  $text = 'This is a test';
  echo strlen($text)."\n";                    //字符串长度为 14,输出 14
  echo substr_count($text,'is')."\n";         //字符串中有两个 is,因此输出 2
  echo substr_count($text,'is',3)."\n";
  //输出 1,从第 4 个位置开始,即'This is a test'中只用 1 个 is
  $text2 = 'gcdgcdgcd';
  echo substr_count($text2,'gcdgcd')."\n";
  //输出 1,重叠的不计算
  echo substr_count($text,'is',3,3)."\n";
  //输出 0,从$text 中的第 4 个字符开始比较而且长度是 3,即's i'中有 0 个'is'
  echo substr_count($text,'is',5,10)."\n";
  //因为 5+10 >14,所以产生警告
?>
```

3. strnatcasecmp()函数

在 PHP 中,strnatcasecmp()函数用于使用“自然顺序”算法比较字符串(不区分大小

写)。该函数的语法格式为:

```
int strnatcasecmp ( string $str1, string $str2 )
```

该函数实现了以人类习惯对数字型字符串进行排序的比较算法。除了不区分大小写,该函数的行为与 strnatcmp()函数类似。

有关参数说明如下。

- $ str1:第一个字符串。
- $ str2:第二个字符串。

返回值:与其他字符串比较函数类似,如果 $ str1 小于 $ str2,返回值小于 0;如果 $ str1 大于 $ str2,返回值大于 0;如果两者相等,返回值为 0。

【示例 33】 使用 strnatcasecmp()函数比较字符串。

eg33.php 代码如下。

```
<?php
  echo "<pre>";
  $st1 = "Wuhan";
  $st2 = "wuhan";
  echo strnatcasecmp ($st1,$st2)."\n";//输出 0,相等
  $st3 = "Wuhan";
  $st4 = "wuhan";
  echo strnatcasecmp ($st3,$st4)."\n";//输出 0,相等,因为不区分大小写
  $st5 = "wuhan is a beautiful city.";
  $st6 = "wuhan";
  echo strnatcasecmp ($st5,$st6)."\n";//输出 1,$st5>$st6
  $st7 = "wuhan";
  $st8 = "wuhh";
  echo strnatcasecmp ($st7,$st8)."\n";//输出-1,第 4 个字符,'a'<'h',按字典中的顺序
?>
```

4. strnatcmp()函数

在 PHP 中,strnatcmp()函数用于使用"自然顺序"算法比较字符串(区分大小写)。该函数的语法格式为:

```
int strnatcmp ( string $str1, string $str2 )
```

该函数实现了以人类习惯对数字型字符串进行排序的比较算法,这就是"自然顺序"。注意比较时区分大小写。

有关参数说明如下。

- $ str1:第一个字符串。
- $ str2:第二个字符串。

返回值:与其他字符串比较函数类似,如果 $ str1 小于 $ str2,返回值小于 0;如果 $ str1 大于 $ str2,返回值大于 0;如果两者相等,返回值为 0。

【示例 34】　使用 strnatcmp()函数比较两个字符串。

eg34.php 代码如下。

```
<?php
  echo "<pre>";
  echo strnatcmp("abc","abc")."\n";     //输出 0,二者相等
  echo strnatcmp("abc","aBc")."\n";     //输出 1,区分大小写,前者大
  echo strnatcmp("900","99")."\n";      //输出 1,当数值比较,前者大
  echo strnatcmp("90","99")."\n";       //输出-1,当数值比较,后者大
?>
```

5. strncasecmp()函数

在 PHP 中,可以使用 strncasecmp()函数来比较两个字符串,该函数与 strcasecmp()函数类似,不同之处在于可以指定两个字符串比较时使用的长度(即最大比较长度)。该函数的语法格式为:

```
int strncasecmp ( string $str1, string $str2, int $len )
```

有关参数说明如下。

- $str1:第一个字符串;
- $str2:第二个字符串;
- $len:最大比较长度。

返回值:如果 str1 小于 str2,返回值小于 0;如果 str1 大于 str2,返回值大于 0;如果两者相等,返回值为 0。

【示例 35】　使用 strncasecmp()函数(该函数不区分大小写)比较两个字符串。

eg35.php 代码如下。

```
<?php
  echo "<pre>";
  $st1 ="abcde";
  $st2 ="abcd";
  echo strncasecmp ($st1,$st2,5)."\n";     //输出 1,前者大,比较长度是 5
  echo strncasecmp ($st1,$st2,4)."\n";     //输出 0,二者相等,比较长度是 4
  $st3 ="aBcd";
  $st4 ="abCde";
  echo strncasecmp ($st3,$st4,5)."\n";     //输出-1,后者大,比较长度是 5
  echo strncasecmp ($st3,$st4,4)."\n";     //输出 0,不区分大小写,比较长度是 4
?>
```

6. strcmp()函数

在 PHP 中,可以使用 strcmp()函数来比较两个字符串。该函数区分大小写,不使用长度。该函数的语法格式为:

```
int strcmp ( string $str1, string $str2 )
```

该函数进行二进制字符串比较,区分大小写。

有关参数说明如下。

- $str1:待比较的第一个字符串。

- $str2：待比较的第二个字符串。

返回值：如果 $str1 小于 $str2，返回值小于 0；如果 $str1 大于 $str2，返回值大于 0；如果两者相等，返回值为 0。该函数区分大小写。

【示例 36】 使用 strcmp()函数（该函数区分大小写，不使用长度）比较两个字符串。

eg36.php 代码如下。

```
<?php
  echo "<pre>";
  $st1 = "abcde";
  $st2 = "abcd";
  echo strcmp($st1,$st2)."\n";//输出 1，前者大
  $st3 = "aBcd";
  $st4 = "abCde";
  echo strcmp($st3,$st4)."\n";//输出 -1，后者大，区分大小写
?>
```

7. similar_text()函数

在 PHP 中，similar_text()函数用于计算两个字符串的相似度，即计算两个字符串的匹配字符的数量。该函数的语法格式为：

```
int similar_text ( string $first, string $second [, float &$percent ] )
```

两个字符串的相似程度计算依据 *Programming Classics：Implementing the World's Best Algorithms by Oliver*（ISBN 0-131-00413-1）一书中的描述进行。注意该实现没有使用 Oliver 虚拟码中的堆栈，但是却进行了递归调用，这个做法可能会导致整个过程变慢或变快。还请注意，该算法的复杂度是 $O(N**3)$，N 是最长字符串的长度。

有关参数说明如下。

- $first：第一个字符串。
- $second：第二个字符串。
- $percent：通过引用方式传递第三个参数，similar_text()将计算相似程度的百分数。

返回值：返回在两个字符串中匹配字符的数目。

【示例 37】 使用 similar_text()函数比较两个字符串的相似度。

eg37.php 代码如下。

```
<?php
  echo "<pre>";
  $var1 = 'Hello';
  $var2 = 'Hello';
  $var3 = 'hello';
  echo similar_text($var1,$var2)."\n";          //5
  echo similar_text($var1,$var3)."\n";          //4
  echo similar_text($var1,$var2,$per1)."\n";    //5,$per1 必须是变量，带回百分百值
  echo similar_text($var1,$var3,$per2)."\n";    //4,$per2 必须是变量，带回百分百值
  echo $per1."\n";                              //100，表示 100%
  echo $per2;                                   //80，表示 80%
?>
```

7.2.8　定位字符串

字符串的定位用于查找子字符串在原字符串中出现的位置和次数，包括第一次出现的位置、最后一次出现的位置，以及出现的总的次数。字符串定位的函数见表 7-8。

表 7-8　字符串定位的函数

函数名称	说　明
stripos()	返回字符串在另一字符串中第一次出现的位置，不区分大小写
stristr()	返回 haystack 字符串从 needle 第一次出现的位置开始到结尾的字符串，不区分大小写
strpos()	返回字符串在另一字符串中第一次出现的位置，区分大小写
strrchr()	查找字符串在另一字符串中最后一次出现的位置
strrev()	反转字符串
strripos()	计算指定字符串在目标字符串中最后一次出现的位置(不区分大小写)
strrpos()	查找字符串在另一字符串中最后一次出现的位置，区分大小写
strspn()	返回字符串中包含特定字符串的个数
strstr()	搜索字符串在另一字符串中首次出现的位置，区分大小写
strchr()	搜索字符串在另一字符串中第一次出现的位置，strstr()函数的别名
strpbrk()	在字符串中搜索指定字符串中的任意一个

上面这些函数用于检索字符串，包括查找某个字符串的第一次、最后一次出现的位置，找某个字符串出现的次数。这些函数区别比较细微。下面将对这些函数进行讲述。

1. stripos()函数

stripos()函数用于查找一个字符串在另一字符串中出现的位置，还可以指定从哪里开始查找，也就是查找的起始位置。该函数对大小写不敏感。该函数的语法格式为：

```
int stripos ( string $haystack, string $needle [, int $offset = 0 ] )
```

有关参数说明如下。

- $haystack：在该字符串中查找。
- $needle：它可以是一个单字符或者多字符的字符串。如果 $needle 不是一个字符串，那么它将被转换为整型并用字符顺序值表示。
- $offset：可选的 $offset 参数用于指定从 $haystack 中的哪个字符开始查找。返回的位置数字值仍然相对于 $haystack 的起始位置。

返回值：返回 $needle 存在于 $haystack 字符串开始的位置(独立于偏移量)。同时注意字符串位置起始于 0，而不是 1。如果未发现 $needle，将返回 false。

【示例 38】 使用 stripos()函数来查找一个字符串在另一字符串中出现的位置。

eg38.php 代码如下。

```
<?php
  echo "<pre>";
  $findme = 'a';
  $mystring1 = 'xyz';
  $mystring2 = 'ABC';
```

```
  $pos1 =stripos($mystring1,$findme);            //false
  $pos2 =stripos($mystring2,$findme);            //0,不区分大小写
  if ($pos1 ===false) {//必须使用"===",因为 0==false 成立,而 0===false 不成立
    echo  "The string '$findme' was not found in the string '$mystring1 '\n";
  }
  if ($pos2 !==false) {//必须使用"!==",因为 0!==false 成立,而 0!=false 不成立
    echo "We found '$findme' in '$mystring2 'at position $pos2\n";
  }
  $pos3 =stripos($mystring2,$findme,1);
  var_dump($pos3);//false,因为查找位置是从第 2 个字符开始的,所以找不到'a'字符
?>
```

2. strrev()函数

在 PHP 中,strrev()函数用于反转字符串。该函数的语法格式为:

```
string strrev ( string $string )
```

说明:$string 参数表示待反转的字符串

返回值:返回反转后的字符串。

【示例 39】 使用 strrev()函数来反转字符串。

eg39. php 代码如下。

```
<?php
  $st1 ="abcdefg";
  $st2 =strrev($st1);
  echo $st2;                                     //输出为 gfedcba
?>
```

3. strripos()函数

在 PHP 中,strripos()函数用于计算指定字符串在目标字符串中最后一次出现的位置(不区分大小写)。该函数的语法格式为:

```
int strripos ( string $haystack, string $needle [, int $offset =0 ] )
```

以不区分大小写的方式查找指定字符串在目标字符串中最后一次出现的位置。与 strrpos()函数不同,strripos()函数不区分大小写。

有关参数说明如下。

- $haystack:在此字符串中进行查找。
- $needle:它可以是一个单字符或者多字符的字符串。
- $offset:该参数可以被指定来查找字符串中任意长度的子字符串。负数偏移表示从字符串的起始位置开始查找,到 offset 位置为止。

返回值:返回 $needle 相对于 $haystack 字符串的位置(与搜索的方向和偏移量无关)。同时注意字符串的起始位置为 0 而非 1。如果 $needle 未被发现,返回 false。

【示例 40】 使用 strripos()函数来查找子字符串最后一次出现的位置。

eg40. php 代码如下。

```
<?php
  echo "<pre>";
```

```
  $haystack = 'ababcd' ;
  $needle = 'aB' ;
  $pos = strripos($haystack,$needle);
  if ($pos === false){
    echo "Sorry, we did not find ($needle) in ($haystack)" ;
  } else {
    echo "Congratulations!\n" ;
    echo "We found the last ($needle) in ($haystack) at position ($pos)\n" ;
  }
  $haystack2 = 'ababcbd' ;
  $needle2 = 'aB' ;
  $pos2 = strripos($haystack2,$needle2,3);  //在'bcbd'中找不到'aB'字符,返回 false
  var_dump($pos2);
?>
```

4. strrpos()函数

在 PHP 中,strrpos()函数用于查找子字符串在字符串中最后一次出现的位置,该函数区分大小写。该函数的语法格式为:

```
int strrpos ( string $haystack, string $needle [, int $offset = 0 ] )
```

注意 PHP 4 中,$needle 只能为单个字符。如果 $needle 被指定为一个字符串,那么仅使用第一个字符。

有关参数说明如下。

- $haystack:在此字符串中进行查找。
- $needle:如果 $needle 不是一个字符串,它将被转换为整型并被视为字符的顺序值。
- $offset:查找字符串中任意长度的子字符串。负数值将导致查找在字符串结尾处开始的计数位置处结束。

返回值:返回 $needle 存在的位置。如果没有找到,返回 false。初始位置为 0 而不是 1。

【示例 41】 使用 strrpos()函数来查找子字符串第一次出现的位置。

eg41.php 代码如下。

```
<?php
  $foo  =  "0123456789a123456789b123456789c" ;
  var_dump ( strrpos ( $foo, '7', -5 ));  //从尾部第 5 个位置开始查找
                                          //结果为 int(17)
  var_dump ( strrpos ( $foo, '7', 20 ));  //从第 20 个位置开始查找
                                          //结果为 int(27)
  var_dump ( strrpos ( $foo, '7', 28 ));  //结果为 bool(false)
?>
```

5. strspn()函数

在 PHP 中,strspn()函数用于计算字符串中全部字符都存在于指定字符集合中的第一段子串的长度。该函数的语法格式为:

```
int strspn ( string $subject, string $mask [, int $start [, int $length ]] )
```

如果省略了＄start 和＄length 参数，则检查整个＄subject 字符串；如果指定了这两个参数，则效果等同于调用＄strspn(substr(＄subject，＄start，＄length)，＄mask)。

有关参数说明如下。

- ＄subject：待检查的字符串。
- ＄mask：检查字符列表。
- ＄start：＄subject 开始检查的位置。如果＄start 被设置并且是非负的，strspn()函数将从＄subject 的第＄start 个位置开始检查。例如，在字符串'abcdef' 中，第 0 个位置的字符是'a'，第二个位置的字符是'c'，等等。如果＄start 被设置并且为负数，strspn()函数将从＄subject 的尾部倒数第＄start 个位置开始检查＄subject。
- ＄length：＄subject 中检查的长度。如果＄length 被设置并且为非负数，那么将从起始位置开始，检查＄subject 的＄length 个长度的字符。如果＄length 被设置并且为负数，那么将从起始位置开始，直到从＄subject 尾部开始第＄length 个位置，对＄subject 进行检查。

返回值：返回第一段全部字符都存在于第二段范围的字符串的长度。

【示例 42】 使用 strspn()函数来返回第一段全部字符都存在于第二段范围的字符串的长度。

eg42.php 代码如下。

```
<?php
  echo "<pre>";
  $var = strspn("423 is the answer to the 128th question.","1234567890");
   //423 在'1234567890'中出现 3 次
  echo $var."\n";
  echo strspn("footff","otf",1,4);
   //打印 4,因为'ootf'在'otf'中出现 4 次
?>
```

6. strpbrk()函数

在 PHP 中，strpbrk()函数用于在字符串中查找一组字符的任何一个字符。该函数的语法格式为：

```
string strpbrk ( string $haystack, string $char_list )
```

strpbrk()函数在＄haystack 字符串中查找＄char_list 中的字符。

有关参数说明如下。

- ＄haystack：在此字符串中查找 char_list。
- ＄char_list：该参数区分大小写。

返回值：返回一个以找到的字符开始的子字符串。如果没有找到，则返回 false 。

【示例 43】 使用 strpbrk()函数在字符串中查找一组字符的任何一个字符。

eg43.php 代码如下。

```
<?php
  echo "<pre>";
  $text = 'This is a Simple text.' ;
```

```
    //输出 "is is a Simple text.",因为'i' 先被匹配
    echo strpbrk($text,'mi')."\n";
    //输出 "Simple text.",因为字符区分大小写
    echo strpbrk($text,'S');
?>
```

7.2.9　替换字符串

使用文字编辑软件的读者可能会感受到 Office 中文本查找和替换的快捷性和准确性。在 PHP 中提供了一些函数,可以用于实现类似办公软件中的查找和替换功能。这些函数见表 7-9。

表 7-9　字符串替换函数

函数名称	说　明
str_ireplace()	替换字符串中的一些字符,大小写不敏感
str_replace()	替换字符串中的一些字符,大小写敏感
substr_replace()	把字符串的一部分替换为另一个字符串
strtr()	将字符串中的指定字符进行替换

1. str_ireplace()函数

该函数返回一个字符串或者数组。该字符串或数组是将 $subject 中全部的 $search 都被 $replace(忽略大小写)替换之后的结果。该函数的语法格式为:

```
mixed str_ireplace ( mixed $search, mixed $replace, mixed $subject [, int &$count])
```

该函数返回一个字符串或者数组。如果没有一些特殊的替换规则,则应该使用该函数替换带有 i 修正符的 preg_replace()函数。

有关参数说明如下。

- $search:要搜索的值,就像是 $needle。可以使用 $array 来提供多个 $needle。
- $replace:使用替换值 $replace 替换查找值 $search,一个数组可以指定多个替换。
- $subject:要被搜索和替换的字符串或数组,就像是 $haystack。如果 $subject 是一个数组,替换操作将遍历整个 $subject,并且也将返回一个数组。
- $count:如果设定了,将会设置执行替换的次数。

提示: 如果 $search 和 $replace 为数组,那么 $str_replace()将对 $subject 做二者的映射替换。如果 $replace 的值的个数少于 $search 的个数,多余的替换将使用空字符串。如果 $search 是一个数组而 $replace 是一个字符串,那么 $search 中每个元素的替换将始终使用这个字符串。如果 $search 或 $replace 是数组,它们的元素将从头到尾一个个处理。

返回值:返回替换后的字符串或者数组。

【示例 44】 字符串的查找和替换,要求使用 str_ireplace()函数。

eg44.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
```

```
    echo "<pre>";
    $subject1 ="Beijing is a big city,Beijing is a beautiful city,I like Beijing.";
    $st1 =str_ireplace("Beijing","Shanghai",$subject1);
    echo $st1."\n";
    //下面的带 count 参数
    $subject2 ="Beijing is a big city,Beijing is a beautiful city,I like Beijing.";
    $st2 =str_ireplace("beijing","Shanghai",$subject2,$count);//不分区大小写
    echo $st2."\n";
    echo $count."\n";
    //使用数组
    $st2 =str_ireplace(array('关胜','林冲','呼延灼','秦明','董平'),array('关羽','张飞','赵
云','马超','黄忠'),'三国五虎将是关胜、林冲、呼延灼、秦明、董平。');
    echo $st2."\n";                        //五虎将全部替换
    $st3 =str_ireplace(array('关胜','林冲','呼延灼','秦明','董平'),array('关羽','张飞','赵
云','马超'),'三国五虎将是关胜、林冲、呼延灼、秦明、董平。');
    echo $st3."\n";                        //'董平'用空字符替换
    $st4 =str_ireplace(array('关胜','林冲','呼延灼','秦明'),array('关羽','张飞','赵云','马
超','黄忠'),'三国五虎将是关胜、林冲、呼延灼、秦明、董平。');
    echo $st4."\n";                        //'董平'不替换
?>
```

str_replace()函数与 str_ireplace()函数类似,不同之处在于前者是区分大小写的,故在此不再赘述。

2. substr_replace()函数

该函数用于把字符串的一部分用另外一个字符串替换。该函数的语法格式为:

```
mixed substr_replace ( mixed $string, mixed $replacement, mixed $start [, mixed $length ] )
```

substr_replace()在字符串 $string 的副本中将由 $start 和可选的 $length 参数限定的子字符串使用 $replacement 进行替换。

有关参数说明如下。

- $string:输入字符串。也可以是一个数组。
- $replacement:替换字符串。
- $start:如果 $start 为正数,替换将从 $string 的 $start 位置开始;如果 $start 为负数,替换将从 $string 的倒数第 $start 个位置开始。
- $length:如果设定了这个参数并且为正数,表示 $string 中被替换的子字符串的长度;如果设定为负数,表示待替换的子字符串结尾处距离 $string 末端的字符个数;如果没有提供此参数,那么它默认为 strlen($string)(字符串的长度)。当然,如果 $length 为 0,那么这个函数的功能为将 $replacement 插入 $string 的 $start 位置处。

返回值:返回结果字符串。如果 $string 是个数组,那么也将返回一个数组。

【示例 45】 使用 substr_replace()函数进行字符串子串的替换。

eg45. php 代码如下。

```
<?php
    echo "<pre>";
```

```
    $var = 'ABCDEFGH:/MNRPQR/' ;
    //这两个例子使用"bob"替换整个$var,即替换位置是 0 到最后,因此输出 bob
    echo substr_replace($var,'bob',0)."\n" ;
    echo substr_replace($var,'bob',0,strlen($var))."\n";
    //将"bob"插入$var 的开头处,因此输出 bobABCDEFGH:/MNRPQR/
    echo substr_replace($var,'bob',0,0)."\n";
    //下面两个例子使用"bob"替换$var 中的"MNRPQR",替换位置是 10 到倒数第 1 或者倒数第 7 到倒
      数第 1,因此输出 ABCDEFGH:/bob/
    echo substr_replace($var,'bob',10,-1)."\n";
    echo substr_replace($var,'bob',-7,-1)."\n";
    //从$var 中删除"MNRPQR",用空字符替换,因此输出 ABCDEFGH://
    echo substr_replace($var,'',10,-1)."\n" ;
    //下面两语句将数组中的全部'XXX'都用'YYY'代替,implode 用于将数组连接成字符串,因此输出
      A: YYY; B: YYY; C: YYY
    $input =array('A: XXX','B: XXX','C: XXX');
    echo implode('; ',substr_replace($input,'YYY',3,3))."\n" ;
    //下面两语句将数组$input 中对应位置的'XXX'分别用$replace 中对应位置的,'AAA','BBB'和'
      CCC'替换,因此输出"A: AAA; B: BBB; C: CCC"
    $replace =array('AAA','BBB','CCC');
    echo implode('; ',substr_replace($input,$replace,3,3))."\n" ;
    //替换的长度来自数组,将 X 变成 AAA,将 XX 变成 BBB,将 XXX 变成 CCC,因此输出"A: AAAXX; B:
      BBBX; C: CCC"
    $length =array(1,2,3);
    echo implode('; ',substr_replace($input,$replace,3,$length));
?>
```

3. strtr()函数

在 PHP 中,strtr()函数用于把字符串中指定的字符转换,但是返回的是副本,原字符串并不会改变。strtr()函数还可以使用数组。该函数的语法格式为:

```
string strtr ( string $str, string $from, string $to )
string strtr ( string $str, array $replace_pairs )
```

该函数返回 $str 的一个副本,并将在 $from 中指定的字符转换为 $to 中相应的字符。比如,$from[$n]中每次的出现都会被替换为 $to[$n],其中 $n 是两个参数都有效的位移($offset)。如果 $from 与 $to 长度不相等,那么多余的字符部分将被忽略。 $str 的长度将会和返回的值一样。

有关参数说明如下。

- $str:待转换的字符串。
- $from:字符串中与将要被转换的目的字符 $to 相对应的源字符。
- $to:字符串中与将要被转换的字符 $from 相对应的目的字符。
- $replace_pairs:该参数可以用来取代 $to 和 $from 参数,因为它是以 array(' from' => 'to', ...)格式出现的数组。

返回值:返回转换后的字符串。如果 $replace_pairs 中包含一个空字符串("")键,那么将返回 false。

【示例 46】 使用 strtr()函数进行字符串子串的替换。

eg46. php 代码如下。

```
<?php
  echo "<pre>";
  $addr1 ="wuhan??chang?sha?";
  //按位置对应:'u'用'1'替换,'h'用'2'替换,'?'用'4'替换,'a'用'5'替换,'g'用空字符替换,因此
    $addr2 值为 w125n44c25ng4s254
  $addr2 =strtr($addr1,"uh??ag","12345");
  echo $addr1."\n";                    //$addr1 保持不变
  echo $addr2."\n";
  $trans =array("hello"=>"hi","hi"=>"hello");
  //'hi'变成'hello','hello'变成'hi',因此输出"hello all, I said hi"
  echo strtr("hi all, I said hello",$trans)."\n";
  //按位置对应,'a'变成'0','b'变成'1',因此输出 1001
  echo strtr("baab","ab","01"),"\n" ;
  $trans =array("ab"=>"01");
  //数组方式,将'ab'变成'01',因此输出 ba01
  echo strtr("baab",$trans);
?>
```

7.2.10 字符串与 HTML 转换

在前面的章节中细心的读者可能注意到了,执行语句"echo "
";"可以实现换行效果,而执行语句"echo "\n";"则不能达到换行效果。PHP 中提供了一些函数用于将字符串中的转义字符转换为 HTML 标记。表 7-10 给出了 HTML 转换函数。

表 7-10 HTML 转换函数

函数名称	说明
hebrev()	把文本从右至左的流转化为从左至右的流
hebrevc()	把文本从右至左的流转化为从左至右的流,同时把\n 转换为
html_entity_decode()	把 HTML 实体转化为字符
htmlentities()	把字符转换为 HTML 实体
htmlspecialchars_decode()	把一些预定义的 HTML 实体转化为字符
htmlspecialchars()	把一些预定义的字符转换为 HTML 实体
nl2br()	把字符串\n 转化为
get_html_translation_table()	用于对文本进行转换
strip_tags()	去掉一个字符串中的 HTML 和 PHP 代码

1. hebrev()函数和 hebrevc()函数

hebrev()函数和 hebrevc()函数用于将逻辑顺序的希伯来文(logical-Hebrew)转换为视觉顺序的希伯来文(visual-Hebrew)。hebrevc()函数还能转换换行符。这两个函数的语法格式如下。

```
string hebrev ( string $hebrew_text [, int $max_chars_per_line =0 ] )
string hebrevc ( string $hebrew_text [, int $max_chars_per_line =0 ] )
```

这两个函数将逻辑顺序希伯来文(logical-Hebrew)转换为视觉顺序希伯来文(visual-Hebrew)。其中 hebrevc()函数还能转换换行符。函数将会尝试避免破坏单词。只有

ASCII 码为 224～251 的 ASCII 字符才受影响。

有关参数说明如下。

- $hebrew_text：逻辑顺序希伯来文字符串。
- $max_chars_per_line：可选参数，表示每行可返回的最多字符数。

返回值：返回视觉顺序的字符串。

【示例 47】 hebrev()函数和 hebrevc()函数的应用举例。

eg47.php 代码如下。

```
<?php
  echo hebrev("? ?? ??? ?????\n");          //输出"????? ??? ?? ?",不能换行
  echo hebrev("abcdefg")."<br />";          //输出 abcdefg
  echo hebrevc("? ?? ??? ?????\n");         //输出"????? ??? ?? ?",能换行
  echo hebrev("abcdefg");                   //输出 abcdefg
?>
```

2. htmlentities()函数

htmlentities()函数用于把字符转换为 HTML 实体。该函数的语法格式为：

```
string htmlentities ( string $string [, int $flags = ENT_COMPAT | ENT_HTML401 [, string
$encoding = 'UTF-8' [, bool $double_encode = true ]]] )
```

本函数用于将字符转换为 HTML 实体。

有关参数说明如下。

- $string：必需。规定要转换的字符串。
- $flags：可选。规定如何处理引号、无效的编码以及使用哪种文档类型。可用的引号类型如下。
  - ✍ENT_COMPAT　默认值。仅编码双引号。
  - ✍ENT_QUOTES　编码双引号和单引号。
  - ✍ENT_NOQUOTES　不编码任何引号。
  - ✍ENT_IGNORE　忽略无效的编码，而不是让函数返回一个空的字符串。应尽量避免，因为这可能对安全性有影响。
  - ✍ENT_SUBSTITUTE　把无效的编码替代成一个指定的带有 Unicode 替代字符 U+FFFD(UTF-8)或者"&#FFFD;"的字符，而不是返回一个空的字符串。
  - ✍ENT_DISALLOWED　把指定文档类型中的无效代码点替代成 Unicode 替代字符 U+FFFD(UTF-8)或者"&#FFFD;"。

规定使用的文档类型的附加 flags 如下。

✍ENT_HTML401　默认值。作为 HTML 4.01 处理代码。

✍ENT_HTML5　作为 HTML 5 处理代码。

✍ENT_XML1　作为 XML 1 处理代码。

✍ENT_XHTML　作为 XHTML 处理代码。

- $character-set：可选。一个规定了要使用的字符集的字符串。允许的值如下。
  - ✍UTF-8　默认值。ASCII 兼容多字节的 8 位 Unicode。
  - ✍ISO-8859-1　西欧规定的字符集。

✍ISO-8859-15　西欧规定的字符集(加入欧元符号＋ISO-8859-1 中丢失的法语和芬兰语字母)。

✍cp866　DOS 专用 Cyrillic 字符集。

✍cp1251　Windows 专用 Cyrillic 字符集。

✍cp1252　Windows 专用西欧字符集。

✍KOI8-R　俄语规定的字符集。

✍BIG5　繁体中文,主要在我国台湾地区使用的字符集。

✍GB 2312　简体中文,我国国家标准字符集。

✍BIG5-HKSCS　主要在我国香港地区使用的字符集。

✍Shift_JIS　日语字符集。

✍EUC-JP　日语字符集。

✍MacRoman　Mac 操作系统使用的字符集。

提示:在 PHP 5.4 之前的版本无法被识别的字符集将被忽略并由 ISO-8859-1 替代。自 PHP 5.4 起,无法被识别的字符集将被忽略并由 UTF-8 替代。

- $double_encode:可选。布尔值,规定是否为编码已存在的 HTML 实体。参数如下。

  ✍true　默认值。将对每个实体进行转换。

  ✍false　不会对已存在的 HTML 实体进行编码。

- $double_encode:当 $double_encode 关闭后,PHP 将不再转换 HTML 实体,默认是 true,表示转换一切。

返回值:返回被转换的字符串。如果 string 包含无效的编码,则返回一个空的字符串,除非设置了 ENT_IGNORE 或者 ENT_SUBSTITUTE 标志。

【示例 48】　htmlentities()函数的应用举例。

eg48.php 代码如下。

```
<?php
  $str = "Bill & 'Steve'";
  echo htmlentities($str, ENT_COMPAT);        //只转换双引号
  echo "<br>";
  echo htmlentities($str, ENT_QUOTES);        //转换双引号和单引号
  echo "<br>";
  echo htmlentities($str, ENT_NOQUOTES);      //不转换任何引号
?>
```

本程序代码的 HTML 输出如下(查看源代码):

```
<!DOCTYPE html>
<html>
<body>
Bill & 'Steve'<br>
Bill & &#039;Tarzan&#039;<br>
Bill & 'Steve'
</body>
</html>
```

3. html_entity_decode()函数

html_entity_decode()函数用于将全部的 HTML 实体转换为字符。此函数的语法格式为：

```
string html_entity_decode ( string $string [, int $flags = ENT_COMPAT | ENT_HTML401 [, string
$encoding = 'UTF-8' ]] )
```

该函数用于将全部 HTML 实体转换为字符，作用与 htmlentities()函数相反。

有关参数说明如下。

- $string：必需。规定要解码的字符串。
- $flags：可选。规定如何处理引号以及使用哪种文档类型。可用的引号类型参考 htmlentities()函数的对应参数。

返回值：返回已转换的字符串。

【示例 49】　html_entity_decode()函数的应用举例。

eg49.php 代码如下。

```
<?php
  $str = "Bill &amp; &#039;Steve&#039;";
  echo html_entity_decode($str);
?>
```

4. htmlspecialchars()函数

htmlspecialchars()函数用于把一些预定的特色字符转换为 HTML 实体，该函数的语法格式为：

```
string htmlspecialchars ( string $string [, int $flags = ENT_COMPAT | ENT_HTML401 [, string $
encoding = 'UTF-8' [, bool $double_encode = true ]]] )
```

本函数仅适用于转换指定的特色字符，包括将“&”转换为“&amp”，“""”转换为“"”，“<”转换为“<”，“>”转换为“>”，等等。

有关参数说明如下。

- $string：要转换的字符。
- $flags：与前面 htmlentities()函数中对应的参数相同。
- $encoding：与前面 htmlentities()函数中对应的参数相同。
- $double_encode：与前面 htmlentities()函数中对应的参数相同。

返回值：返回转换后的字符串。

【示例 50】　htmlspecialchars()函数的应用举例。

eg50.php 代码如下。

```
<?php
  $new = htmlspecialchars ( "<a href='test'>Test</a>" , ENT_QUOTES );
  echo $new ;//“&lt;a href=&#039;test&#039;&gt;Test&lt;/a&gt;”,在源代码中查看
?>
```

5. htmlspecialchars_decode()函数

htmlspecialchars_decode()函数将特殊的 HTML 实体转换回普通字符。该函数的语

法格式为：

```
string htmlspecialchars_decode ( string $string [, int $flags = ENT_COMPAT | ENT_HTML401 ] )
```

此函数的作用和 htmlspecialchars()函数刚好相反，它将特殊的 HTML 实体转换回普通字符。被转换的实体有“&”“ "”(没有设置 ENT_NOQUOTES 时)、“'”(设置了 ENT_QUOTES 时)、“<” 以及“>”。

有关参数说明如下。

- $ string：要解码的字符串。
- $ flags：用下列标记中的一个或多个作为一个位掩码，来指定如何处理引号和使用哪种文档类型。默认为 ENT_COMPAT | ENT_HTML401。有效的 flags 常量如下。
  - ENT_COMPAT　转换双引号，不转换单引号。
  - ENT_QUOTES　单引号和双引号都转换。
  - ENT_NOQUOTES　单引号和双引号都不转换。
  - ENT_HTML401　作为 HTML 4.01 编码处理。
  - ENT_XML1　作为 XML 1 编码处理。
  - ENT_XHTML　作为 XHTML 编码处理。
  - ENT_HTML5　作为 HTML 5 编码处理。

返回值：返回解码后的字符串。

【示例 51】 使用 htmlspecialchars_decode()函数进行 HTML 实体转换。

eg51.php 代码如下。

```
<?php
  $str = "<p>this -&gt; &quot;</p>\n";
  echo htmlspecialchars_decode($str);
  //注意，这里的引号不会被转换
  echo htmlspecialchars_decode($str,ENT_NOQUOTES);
?>
```

输出：

```
this ->"

this ->"
```

说明：从输出结果可以看出，“>”表示大于号，“"”表示半角双引号，而<p>和</p>标记则是换行符。

6. get_html_translation_table()函数

get_html_translation_table()函数返回使用 htmlspecialchars()函数和 htmlentities()函数后的转换表。该函数的语法格式为：

```
array get_html_translation_table ([ int $table = HTML_SPECIALCHARS [, int $flags =
ENT_COMPAT | ENT_HTML401 [, string $encoding = 'UTF-8' ]]] )
```

get_html_translation_table()函数将返回 htmlspecialchars()函数和 htmlentities()函

数处理后的转换表。特殊字符可以使用多种转换方式。例如："""可以被转换成""""""或者"&#x22"，get_html_translation_table()函数返回其中最常用的。

有关参数说明如下。

- $table：有两个新的常量(HTML_ENTITIES，HTML_SPECIALCHARS)允许指定想要的表。
- $flags：可选。规定翻译表将包含哪种引号以及翻译表用于哪种文档类型。可用的引号类型如下。

  ✍ENT_COMPAT　默认值。翻译表包含双引号实体,不包含单引号实体。

  ✍ENT_QUOTES　翻译表包含双引号实体和单引号实体。

  ✍ENT_NOQUOTES　翻译表不包含双引号实体和单引号实体。

规定翻译表适用的文档类型的附加 flags 如下。

✍ENT_HTML401　默认值。HTML 4.01 的翻译表。

✍ENT_HTML5　HTML 5 的翻译表。

✍ENT_XML1　XML 1 的翻译表。

✍ENT_XHTML　XHTML 的翻译表。

- $encoding：用于编码。如果省略则默认为编码使用。如果省略,默认值是 ISO-8859-1。

返回值：将转换表作为一个数组返回。

【示例 52】　get_html_translation_table()函数的应用举例。

eg52.php 代码如下。

```
<?php
  echo "<pre>";
  var_dump(get_html_translation_table(HTML_ENTITIES,ENT_QUOTES|ENT_HTML5));
?>
```

程序运行结果如下。

```
array(1511) {
  ["	"]=>
  string(5) "	"
  ["
"]=>
  string(9) "
"
  ["!"]=>
  string(6) "!"
  ["""]=>
  string(6) """
  ["#"]=>
  string(5) "#"
  ["$"]=>
  string(8) "$"
  ["% "]=>
  string(8) "% "
```

```
  ["&"]=>
  string(5) "&"
}
```

7. strip_tags()函数

使用 strip_tags()函数可从字符串中去除 HTML 和 PHP 标记。该函数的语法格式为：

```
string strip_tags ( string $str [, string $allowable_tags ] )
```

该函数尝试返回给定的字符串 $str，是去除空字符、HTML 和 PHP 标记后的结果。它使用与 fgetss()函数一样的机制删除标记。

有关参数说明如下。

- $str：输入字符串。
- $allowable_tags：使用可选的第二个参数指定不被删除的字符列表。HTML 注释和 PHP 标签也会被删除。这里是硬编码处理的，所以无法通过 allowable_tags 参数进行改变。

返回值：返回处理后的字符串。

【示例 53】 strip_tags()函数的应用举例。

eg53.php 代码如下。

```
<?php
  $text = '<p>Test paragraph.</p><!-- Comment --><a href="#fragment">Other text</a>';
  echo strip_tags($text);
  echo "\n";
  //允许<p>和<a>标记
  echo  strip_tags($text,'<p><a>');
?>
```

程序运行结果如图 7-7 所示。



图 7-7 strip_tags()函数的程序运行结果

说明：在运行结果中可见段落<p>和超链接<a>。

7.3　字符串编码

在 PHP 中整型、浮点型、日期时间型数据的格式都是固定的，在内存中存储的单元大小也是固定的。在读/写过程中不会出现问题。但是对于字符串类型，它不仅可以包含汉字、英文和数字，甚至还可以包含其他语言文字，因此准确地存储、读出及其他处理字符串就显得尤为重要。本节将介绍有关字符编码的相关知识。

7.3.1　字符集与编码

计算机中储存的信息都是用二进制数表示的，而我们在屏幕上看到的英文、汉字等字符是二进制数转换之后的结果。通俗地说，按照何种规则将字符存储在计算机中，如'a'用什么表示，称为编码；反之，将存储在计算机中的二进制数解析显示出来，称为解码，如同密码学中的加密和解密。在解码过程中如果使用了错误的解码规则，则导致'a'解析成'b'或者乱码。

字符集(Charset)是一个系统支持的所有抽象字符的集合。字符是各种文字和符号的总称，包括各国的文字、标点符号、图形符号、数字等。

字符编码(Character Encoding)是一套法则，使用该法则能够对自然语言的字符的一个集合(如字母表或音节表)，与其他东西的一个集合(如号码或电脉冲)进行配对。即在符号集合与数字系统之间建立对应关系，它是信息处理的一项基本技术。通常人们用符号集合(一般情况下就是文字)来表达信息。而以计算机为基础的信息处理系统则是利用元件(硬件)不同状态的组合来存储和处理信息的。元件不同状态的组合能代表数字系统的数字，因此字符编码就是将符号转换为计算机可以接收的数字系统的数，称为数字代码。

常见字符集名称有 ASCII 字符集、GB 2312 字符集、BIG5 字符集、GB 18030 字符集、Unicode 字符集等。计算机要准确地处理各种字符集文字，需要进行字符编码，以便计算机能够识别和存储各种文字。

1. ASCII 字符集

ASCII 码是目前计算机中用得最广泛的字符集及其编码，是由美国国家标准局(ANSI)制定的 ASCII 码(American Standard Code for Information Interchange，美国标准信息交换码)，它已被国际标准化组织(ISO)定为国际标准，称为 ISO 646 标准。适用于所有拉丁文字字母，ASCII 码有 7 位码和 8 位码两种形式。最多可以给 256 个字符(包括字母、数字、标点符号、控制字符及其他符号)分配(或指定)数值。

ASCII 码于 1968 年提出，用于在不同计算机硬件和软件系统中实现数据传输标准化，在大多数的小型机和全部的个人计算机都使用此码。

基本的 ASCII 字符集共有 128 个字符，其中有 96 个可打印字符，包括常用的字母、数字、标点符号等，另外还有 32 个控制字符。标准 ASCII 码使用 7 个二进位对字符进行编码，对应的 ISO 标准为 ISO 646 标准。

ASCII 字符集和编码根据二进制计数法，7 位二进制数可以表示 128(2^7)种状态，每种状态都唯一地编为一个 7 位的二进制码(0000000～1111111)，对应一个字符(或控制码)，这

些码可以排列成一个十进制序号 0～127。所以,7 位 ASCII 码是用 7 位二进制数进行编码的,可以表示 128 个字符。

其中第 0～32 号及第 127 号(共 34 个)是控制字符或通信专用字符,如 LF(换行)、CR(回车)、FF(换页)、DEL(删除)、BS(退格)、BEL(振铃)等控制符;通信专用字符包括 SOH(文头)、EOT(文尾)、ACK(确认)等;

第 33～126 号(共 94 个)是字符,其中第 48～57 号为 0～9 十个阿拉伯数字;第 65～90 号为 26 个大写英文字母,第 97～122 号为 26 个小写英文字母,其余为一些标点符号、运算符号等。

注意:字母和数字的 ASCII 码的记忆是非常简单的。我们只要记住了一个字母或数字的 ASCII 码(例如记住 A 为 65,0 的 ASCII 码为 48),知道相应的大小写字母之间差 32,就可以推算出其余字母、数字的 ASCII 码。

虽然标准 ASCII 码是 7 位编码,但由于计算机基本处理单位为字节(1byte＝8bit),所以一般仍以一个字节来存放一个 ASCII 字符。每一个字节中多余出来的一位(最高位)在计算机内部通常保持为 0 (在数据传输时可用作奇偶校验位)。

2. GB 2312 字符集

GB 2312 码是中华人民共和国国家汉字信息交换用编码,全称为《信息交换用汉字编码字符集——基本集》(GB 2312—1980),由国家标准总局发布,1981 年 5 月 1 日实施,通行于中国大陆。新加坡等地也使用此编码。

GB 2312 码是计算机可以识别的编码,适用于汉字处理、汉字通信等系统之间的信息交换。基本集共收入汉字 6763 个和非汉字图形字符 682 个。整个字符集分成 94 个区,每区有 94 个位,每个区位上只有一个字符,因此可用所在的区和位来对汉字进行编码,称为区位码。

这个码是唯一的,不会有重码字。把换算成十六进制的区位码加上 2020H,就得到国标码。国标码加上 8080H,就得到常用的计算机机内码。1995 年又颁布了《汉字编码扩展规范》(GBK)。GBK 与 GB 2312—1980 国家标准所对应的内码标准兼容,同时在字汇一级支持 ISO/IEC 10646—1 和 GB 13000—1 的全部中国、日本、韩国(CJK)汉字,共计 20902 字。信息交换用汉字编码字符集和汉字输入编码之间的关系是,根据不同的汉字输入方法,通过必要的设备向计算机输入汉字的编码,计算机接收之后,先转换成信息交换用汉字编码字符,这时计算机就可以识别并进行处理;汉字输出是先把机内码转换成汉字编码,再发送到输出设备。

GB 2312 的出现,基本满足了汉字的计算机处理需要,它所收录的汉字已经覆盖中国大陆 99.75%的使用频率。

对于人名、古汉语等方面出现的罕用字,GB 2312 不能处理,这导致了后来 GBK 及 GB 18030 汉字字符集的出现。

GB 2312 中对所收汉字进行了“分区”处理,每区含有 94 个汉字/符号。

- 01～09 区为特殊符号。
- 16～55 区为一级汉字,按拼音排序。
- 56～87 区为二级汉字,按部首/笔画排序。
- 10～15 区及 88～94 区则未编码。

举例来说，“啊”字是 GB 2312 中 16 区的第一个汉字，它的区位码就是 1601。

3. BIG5 字符集

BIG5 码是通行于我国台湾、香港地区的一个繁体字编码方案，俗称“大五码”。它并不是一个法定的编码方案，存在着一些瑕疵，业界的评价也不高，但它广泛地被应用于计算机业，尤其是国际互联网中，从而成为一种事实上的行业标准。

BIG5 码是 1984 年我国台湾资讯工业策进会根据《通用汉字标准交换码》制订的编码方案。

BIG5 是一个双字节编码方案，其第一字节的值在十六进制的 A0～FE 之间，第二字节在 40～7E 和 A1～FE 之间。因此，其第一字节的最高位是 1，第二字节的最高位则可能是 1，也可能是 0。

BIG5 码的图形符号及汉字基本与 CNS 11643 标准的第一、第二字面（Plane）一致，它收录 13461 个符号和汉字，包括：

- 符号 408 个，编码位置为 A140～A3FE(实际止于 A3BF，末尾有空白位置)。
- 汉字 13053 个，分为常用字和次常用字两部分，各部分中的汉字按笔画/部首排列。其中常用字为 5401 个，编码位置为 A440～C67E；次常用字为 7652 个，编码位置为 C940～F9FE(实际止于 F9D5，末尾有空白位置)。

其余的 A040～A0FE、C6A1～C8FE、FA40～FEFE 为空白区域。一些空白位置经常被用于用户造字区。

4. GB 18030 字符集

（1）名称的由来

GB 18030 的全称是《信息交换用汉字编码字符集基本集的扩充》(GB 18030—2000)，是我国政府于 2000 年 3 月 17 日发布的新的汉字编码国家标准，2001 年 8 月 31 日后在中国市场上发布的软件必须符合本标准。

（2）特点

GB 18030 字符集标准的出台经过广泛参与和论证，来自国内外知名信息技术行业的公司，信息产业部和原国家质量技术监督局联合实施。

GB 18030 字符集标准解决了汉字、日文假名、朝鲜语和中国少数民族文字组成的大字符集计算机编码问题。该标准的字符总编码空间超过 150 万个编码位，收录了 27484 个汉字，覆盖中文、日文、朝鲜语和中国少数民族文字。满足中国、日本和韩国等东亚地区信息交换多文种、大字量、多用途、统一编码格式的要求。与 Unicode 3.0 版本兼容，填补了 Unicode 扩展字符字汇“统一汉字扩展 A”的内容。并且与以前的国家字符编码标准(GB 2312、GB 13000.1)兼容。

（3）编码方法

GB 18030 标准采用单字节、双字节和 4 字节 3 种方式对字符编码。单字节部分使用 0×00～0×7F 码(对应于 ASCII 码的相应码)。双字节部分，首字节码从 0×81～0×FE，尾字节码位分别是 0×40～0×7E 和 0×80～0×FE。4 字节部分采用 GB/T 11383 未采用的 0×30～0×39 作为对双字节编码扩充的后缀，这样扩充的 4 字节编码，其范围为 0×81308130～0×FE39FE39。其中第 1、3 字节编码码位均为 0×81～0×FE，第 2、4 字节编码码位均为 0×30～0×39。

（4）包含的内容

双字节部分收录内容主要包括 GB 13000.1 全部 CJK 汉字 20902 个、有关标点符号、表意文字描述符 13 个、增补的汉字和部首/构件 80 个、双字节编码的欧元符号等。

4 字节部分收录了上述双字节字符之外的，包括 CJK 统一汉字扩充 A 在内的 GB 13000.1 中的全部字符。

5．unicode 字符集

最初的 unicode 编码是固定长度的，共 16 位，也就是 2 字节代表一个字符，这样一共可以表示 65536 个字符。显然，这样要表示各种语言中所有的字符是远远不够的。unicode 4.0 规范考虑到了这种情况，定义了一组附加字符编码，附加字符编码采用 2 个 16 位来表示，这样最多可以定义 1048576 个附加字符，目前 unicode 4.0 只定义了 45960 个附加字符。

unicode 只是一个编码规范，目前实际实现的 unicode 编码主要有 3 种：UTF-8、UCS-2 和 UTF-16，3 种 unicode 字符集之间可以按照规范进行转换。

（1）UTF-8

UTF-8 是一种 8 位的 unicode 字符集，编码长度是可变的，并且是 ASCII 字符集的严格超集，也就是说 ASCII 中每个字符的编码在 UTF-8 中是完全一样的。UTF-8 字符集中，一个字符可能是 1～4 字节长。一般来说，欧洲的字母字符长度为 1～2 字节，而亚洲的大部分字符则是 3 字节，附加字符为 4 字节长。

UNIX 平台中普遍支持 UTF-8 字符集，HTML 和大多数浏览器也支持 UTF-8，而 Windows 和 Java 则支持 UCS-2。

UTF-8 的主要优点如下。

- 对于欧洲字母及字符需要较少的存储空间。
- 容易从 ASCII 字符集向 UTF-8 迁移。

（2）UCS-2

UCS-2 是固定长度为 16 位的 unicode 字符集。每个字符都是 2 字节，UCS-2 只支持 unicode 3.0，所以不支持附加字符。

UCS-2 的优点如下。

- 对于亚洲字符的存储空间需求比 UTF-8 少，因为每个字符都是 2 字节。
- 处理字符的速度比 UTF-8 更快，因为是固定长度编码的。
- 对于 Windows 和 Java 的支持更好。

（3）UTF-16

UTF-16 是一种 16 位编码的字符集。实际上，UTF-16 就是 UCS-2 加上附加字符的支持，也就是符合 unicode 4.0 规范的 UCS-2，所以 UTF-16 是 UCS-2 的严格超集。

UTF-16 中的字符，要么是 2 字节，要么是 4 字节。UTF-16 主要在 Windows 2000 以上版本使用。

（4）AL32UTF8

一种 UTF-8 编码的字符集，支持最新的 unicode 4.0 标准。字符长度为 1～3 字节，附加字符则为 4 字节长。

(5) UTF8

支持 unicode 3.0 的 UTF-8 编码方式。由于附加字符是在 unicode 3.1 中提出的，UTF8 不支持附加字符。但是 unicode 3.0 已经为附加字符预留了编码空间，所以即使在 UTF8 的数据库中插入附加字符也是可以的，只是数据库会将该字符分隔成两部分，需要占 6 个字符的长度。所以，如果需要支持附加字符，那么建议将数据库的字符集切换为新的 AL32UTF8。

UTF8 可用于数据库字符集，也可用于国家字符集。

(6) UTFE

UTFE 是基于 EBCDIC 平台的 unicode 字符集，就像 ASCII 平台上的 UTF8 一样。不同的是，UTFE 中每个字符可能占 1～4 字节，而附加字符则需要 8 字节来表示。

(7) AL16UTF16

AL16UTF16 是一种 UTF-16 编码的 unicode 字符集，在 Oracle 中用于国家字符集。

(8) AL24UTFFSS

该字符集只支持 unicode 1.1 规范，在 Oracle 7.2～8i 版本中使用，目前已经淘汰。

7.3.2　页面编码设置

在 PHP 中可以使用 UTF、GBK 和 BIG5 等各种编码格式，可以使用 PHP 函数或者使用 HTML 标记来设置页面编码。

1. header()函数

在 PHP 中可以使用 header()函数来设置页面编码。函数的作用是把括号里面的信息发送到 https 标头，因此通常需要放置在 PHP 页面的首页。其语法格式如下。

```
void header("content-type:text/html;charset=xxx")
```

其中×××就是用户需要设置的网页编码。

例如：

• PHP 页面为 UTF 编码

```
header("Content-type: text/html; charset=utf-8");
```

• PHP 页面为 GBK 编码

```
header("Content-type: text/html; charset=gb2312");
```

• PHP 页面为 BIG5 编码

```
header("Content-type: text/html; charset=big5");
```

通常情况以上代码放在 PHP 页面的首页

2. <meta>标记

网页中还可以使用<meta>标记设置编码。代码为：

```
<META http-equiv="content-type" content="text/html; charset=xxx">
```

其中 xxx 就是要设置的编码格式。

例如：

- 使用<meta>标记将页面设置为 UTF-8 编码格式：

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

- 使用<meta>标记将页面设置为 BIG-5 编码格式：

```
<meta http-equiv="Content-Type" content="text/html; charset=big-5 " />
```

- 使用<meta>标记将页面设置为 GB 2312 编码格式：

```
<meta http-equiv=Content-Type content="text/html; charset=gb2312">
```

- 使用<meta>标记将页面设置为 GB 18030 编码格式：

```
<meta http-equiv="content-type"content="text/html;charset=gb18030">
```

- 使用<meta>标记将页面设置为 GBK 编码格式：

```
<meta http-equiv="Content-Type" content="text/html; charset=gbk" />
```

3. 两种页面编码设置的区别和优先级

https 标头是服务器以 HTTP 协议传送 HTML 信息到浏览器前所送出的字串。因为 meta 标签是属于 HTML 信息的，所以 header()函数发送的内容先到达浏览器，也就是 header()函数的优先级高于 meta。假如一个 PHP 页面既有 header("content-type:text/html; charset=***")，又有<meta http-equiv="content-type" content="text/html; charset=***">，浏览器就只认 https 标头而不认 meta 了。当然这个函数只能在 PHP 页面内使用。

【示例 54】 测试 header()函数设置的编码和使用<meta>标记设置的编码的优先级。

eg54. php 代码如下。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>无标题文档</title>
</head>
<body>
<?php
  header("Content-type: text/html; charset=utf-8");
  $str="你好";
  echo $str,strlen($str);
?>
</body>
</html>
```

在运行该页面，右击并选择“编码”命令，读者会发现页面编码格式是 Unicode(UTF-8)，说明使用 header()函数设置的编码优先。

4. Apache 配置

可以在服务器上设置默认的网页编码。在 Apache 根目录的 conf 文件夹中有一个

Apache 服务器配置文件 httpd.conf，使用如下语句。

```
AddDefaultCharset xxx
```

可以将整个服务器中的网页的编码设置为 xxx，其中 xxx 就是要设置的编码，如 UTF-8 等。在前面加分号可以注释该语句。

该语句的作用相当于在每个网页前面加了一行语句“header("content-type:text/html;charset=xxx");”。如果要想设置的默认编码不起作用，可以在自己的网页前面加上一行“header("content-type:text/html;charset=yyy");”，也就是说页面中用户书写的那行 header 语句是优先于 httpd.conf 文件中的设置的。而<meta>仅在两个 header 语句都没有设置的前提下才发生作用。

这就是为什么有时候网页中明明使用<meta>设置了编码而没有发生作用的原因，此时应该检查一下 Apache\conf\httpd.conf 中是否使用“header("content-type:text/html;charset=xxx");”设置了默认编码。

5. 编码的一致性

在 PHP 中最常用的数据库是 MySQL 数据库，二者常常结合使用，因此数据库的编码要和 PHP 页面编码对应。这就是编码的一致性。为了确保不出现乱码，更要做到 MySQL 数据库编码、HTML 页面编码、HTML 文件编码、后台程序编码(写入数据库)要全部一致。

(1) MySQL 数据库编码：

建立数据库时指定编码(如 utf_general_c)；建立数据表、建立字段、插入数据时不要指定编码，此时会自动继承数据库的编码。

数据库连接时也有编码，可以在连接完数据库后，执行如下语句。

```
mysql_query('SET NAMES utf-8');          //将 UTF-8 换成自己的编码，如 GBK
```

(2) HTML 页面的编码，指的是下面这一行的设置：

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

(3) 需要注意的是，JavaScript 或 Flash 中传递的数据是 UTF-8 编码，如果数据库和页面编码是 GBK，要进行转码，然后写入数据库。

(4) 在后台程序(PHP、Python)中指定源程序的编码：

```
header('Content-type: text/html; charset=utf-8');
```

7.3.3 编码转换

PHP 输出的字符串编码和 header 头信息声明的编码不一致时会输出乱码。PHP 程序开发中，编码问题一定困扰了不少人，比如：当我们需要输出 GBK 编码的字符串时，但不知道传过来的字符串是 GBK 编码还是 UTF-8 编码，因此无法转换编码，这时需要一个能统一编码的函数。

在 PHP 中，通常使用 mb_convert_encoding()函数和 iconv()函数来进行数据的编码转换。

1. mb_convert_encoding()函数

mb_convert_encoding()函数可以指定多种输入编码，它会根据内容自动识别，但是执

行效率比 iconv 差太多。该函数的语法格式为：

```
string mb_convert_encoding ( string $str , string $to_encoding [, mixed $from_encoding =mb_
internal_encoding() ] )
```

mb_convert_encoding()函数功能非常强大，如果能够知道一种字符的编码格式，基本上都可以转换成想要的格式。

有关参数说明如下。

- $str：要编码的字符串；
- $to_encoding：要转换的编码类型；
- $from_encoding：在转换前通过字符代码名称来指定。它可以是一个数组，也可以是逗号分隔的枚举列表。如果没有提供 $from_encoding，则会使用内部（internal）编码。

返回值：编码后的字符串。

【示例 55】 mb_convert_encoding()函数的应用举例。

eg55.php 代码如下。

```
<?php
  header("content-Type: text/html; charset=utf-8");
  $path ="eg55.txt";      //内容为 '武汉',保存格式是 ANSI
  $st =file_get_contents($path);
  echo mb_convert_encoding($st, "UTF-8", "GBK")."<br />";
  //转换为 UTF-8 格式(与 header 一致),正常显示武汉
  echo $st;               //显示乱码
?>
```

程序运行结果如图 7-8 所示。

【示例 56】 mb_convert_encoding()函数的应用举例。

eg56.php 代码如下。

```
<?php
  header("content-Type: text/html; charset=gb2312");
  $path ="eg56.txt";      //内容为 '武汉',保存格式是 ANSI,即 GB 2312 编码
  $st =file_get_contents($path);
  echo mb_convert_encoding($st, "UTF-8", "GBK")."<br />";
  //转换为 UTF-8 格式(与 header 不一致),显示乱码
  echo $st;               //格式与 header 一致,显示'武汉'
?>
```

程序运行结果如图 7-9 所示。



图 7-8　示例 55 的程序运行效果



图 7-9　示例 56 的程序运行效果

2. iconv()函数

在 PHP 中，字符串按要求的字符编码来转换可使用 iconv()函数。该函数的语法格

式为：

```
string iconv ( string $in_charset , string $out_charset , string $str )
```

将字符串 str 从 in_charset 转换编码到 out_charset。

有关参数说明如下。

- $in_charset：输入的字符集。
- $out_charset：输出的字符集。如果在 $out_charset 后添加了字符串“//TRANSLIT”，将启用转写(transliteration)功能。也就是当一个字符不能被目标字符集所表示时，它可以通过一个或多个形似的字符来近似表达。如果添加了字符串“//IGNORE”，不能以目标字符集表达的字符将被默默丢弃。否则，$str 从第一个无效字符开始截断并导致一个错误提示。
- $str：要转换的字符串。

返回值：返回转换后的字符串，或者在失败时返回 false。

【示例 57】 iconv()函数的应用举例。

eg57.php 代码如下。

```
<?php
  header("content-type: text/html; charset=UTF-8");
  $path = "eg57.txt";     //内容为 'PHP 程序设计'，保存格式是 ANSI，即 GB 2312 编码
  $st=file_get_contents($path);  //读出文件内容'PHP 程序设计'
  echo $st."<br />";     //直接显示会出现乱码
  echo iconv("GB2312", "UTF-8", $st)."<br />";
  //转换为 UTF-8 格式(与 header 一致)，显示'PHP 程序设计'
?>
```

3. mb_detect_encoding()函数

检测字符的编码使用 mb_detect_encoding()函数，该函数的语法格式为：

```
string mb_detect_encoding ( string $str [, mixed $encoding_list =mb_detect_order() [, bool
$strict = false ]] )
```

该函数的作用是检测字符串 $str 的编码。

有关参数说明如下。

- $str：待检查的字符串。
- $encoding_list：这是一个字符编码列表。编码顺序可以由数组或者逗号分隔的列表字符串指定。如果省略了 $encoding_list，将会使用 $detect_order。
- $strict：$strict 指定了是否严格地检测编码。默认是 false。

返回值：检测到的字符编码。无法检测指定字符串的编码时，返回 false。

【示例 58】 mb_detect_encoding()的应用举例。先判断文本文件的编码，再转换编码并输出。

eg58.php 代码如下。

```
<?php
  header("content-type: text/html; charset=UTF-8");
  $path = "eg58.txt";     //内容为 'PHP 程序设计'，保存格式是 ANSI，即 GB 2312 编码
```

```
    $st = file_get_contents($path);  //读出文件内容'PHP 程序设计'
    $arr = array("ASCII","GB2312","GBK","UTF-8","EUC-CN","CP936");
    $encode = mb_detect_encoding($st, $arr);
    //如果是 GB 2312 编码,则转换为 UTF-8 编码,ANSI 是一种更严的 GB 2312 编码
    if ($encode == 'EUC-CN'){
       $st = iconv("GB2312", "UTF-8", $st);
       //或用$st = mb_convert_encoding($st, "UTF-8", "GBK");
       //若不转换而直接输出则显示乱码
    }
    echo $st;
?>
```

7.3.4 字符串加密

为了确保信息的安全性,需要用到字符串加密。PHP 提供了一些用于字符串加密和解密的函数,来满足不同的加密和解密需求。多种加密函数的存在是为了满足不同需要以及混淆字符串加密原理,提高破解难度。

字符串加密技术有两种,一种是只接加密,而不提供解密技术的单向加密;另一种是可逆的加密技术,即加密后的数据可以被逆向执行并实现解密。

与加密和解密有关的函数见表 7-11。

表 7-11 字符串加密和解密函数

函数名称	说明
crypt()	单向字符串加密算法
md5()	计算字符串的 MD5 散列
md5_flie()	计算文件的 MD5 散列
sha1()	计算字符串的 SHA-1 散列
sha1_file()	计算文件的 SHA-1 散列
str_shuffle()	随机打乱字符串中的所有字符
base64_encode()	使用 MIME base64 对数据进行编码
base64_decode()	对使用 MIME base64 编码的数据进行解码
urlencode()	编码 URL 字符串
urldecode()	解码已编码的 URL 字符串

1. crypt()函数

在 PHP 中,单向字符串散列 crypt()函数用于加密字符串,该函数的语法格式为:

```
string crypt ( string $str [, string $salt ] )
```

crypt()函数返回一个基于标准 UNIX DES 算法或系统上其他可用的替代算法的散列字符串。有些系统支持不止一种散列类型。实际上,有时候基于 MD5 的算法被用来替代基于标准 DES 的算法,这种散列类型由盐值参数触发。在 PHP 5.3 之前,PHP 在安装时根据系统的 crypt()函数决定可用的算法。如果没有提供盐值,PHP 将自动生成一个 2 个字符(DES)或者 12 个字符(MD5)的盐值,这取决于 MD5 crypt()的可用性。PHP 设置了一个名为 CRYPT_SALT_LENGTH 的常量,用来表示可用散列允许的最长可用盐值。

有关参数说明如下。

- $ str：待散列的字符串。
- $ salt：可选的盐值字符串。如果没有提供，将由不同的算法实现，并可能导致不可预料的结束。

返回值：返回散列后的字符串或一个少于 13 个字符的字符串，从而保证在失败时与盐值区分开。

2. md5()函数

计算字符串的 MD5 散列值。该函数的语法格式为：

```
string md5 ( string $str [, bool $raw_output = false ] )
```

使用 RSA 数据安全公司的 MD5 报文算法计算 $ str 的 MD5 散列值。

有关参数说明如下。

- $ str：原始字符串。
- $ raw_output：如果可选的 raw_output 被设置为 true，那么 MD5 报文摘要将以 16 字节长度的原始二进制格式返回。

返回值：以 32 字符十六进制数字形式返回散列值。

【示例 59】 使用 crypt()函数和 md5()函数进行加密。

eg59.php 代码如下。

```
<?php
  $password = crypt ('mypassword');
  echo "crypt 加密结果:".$password;
  echo "<br />";
  $password = '123abc456fde';   //加密后是"66e557072cefb9ac79f6261b001ab6d8"
  if (md5($password) === '66e557072cefb9ac79f6261b001ab6d8'){
    echo "You are welcome!" ;
  }
?>
```

3. md5_file()函数

md5_file()函数用于对文件进行加密。该函数的语法格式为：

```
string md5_file ( string $filename [, bool $raw_output = false ] )
```

使用 RSA 数据安全公司的 MD5 报文算法计算 filename 文件的 MD5 散列值并返回。该散列值为 32 字符的十六进制数字。

有关参数说明如下。

- $ filename：文件名。
- $ raw_output：如果被设置为 true，那么报文摘要将以原始的 16 位二进制格式返回。

返回值：成功返回字符串，否则返回 false。

【示例 60】 使用 md5_file()函数判断文本文件是否被修改。分为两个程序，第一个程序用于对文本文件进行加密，第二个文件用于判断文件是否被修改。

eg60-1.php 代码如下。

```
<?php
  //加密文件 eg60.txt,得到字符串$md5file
  $md5file =md5_file("eg60.txt");
  //文件加密后的字符串$md5file 再写入 md5file.txt 文件
  file_put_contents("md5file.txt",$md5file);
?>
```

eg60-2.php 代码如下。

```
<?php
  //获得(见 eg60-1.php)加密后文件的内容并赋值给$md5file,它是一个加密后的字符串
  $md5file =file_get_contents("md5file.txt");
  //再次加密原文件所得到的字符串$st
  $st =md5_file("eg60.txt");
  /*将第一次加密文件的内容$md5file 和第二次加密原文件的字符串$st 进行比较,
    若二者相等,说明文件没有被改变 */
  if ($st ==$md5file)
  {
    echo "The file is ok.";
  } else  {
    echo "The file has been changed.";
  }
?>
```

4. sha1()函数

sha1()函数利用美国 Secure Hash 算法 1 求字符串的散列,该函数的语法格式为:

```
string sha1 ( string $string [ ,bool $raw =false ] )
```

该函数利用美国 Hash 算法 1 求字符串的散列。

有关参数说明如下。

- $string:必需。规定要计算的字符串。
- $raw:可选。规定十六进制或二进制输出格式。true 表示 20 字符的二进制格式,false 为默认值,表示 40 字符的十六进制数。

返回值:如果成功,则返回已计算的 SHA-1 散列;如果失败,则返回 false。

【示例 61】 使用 sha1()函数进行字符串加密。

eg61.php 代码如下。

```
<?php
  echo "<pre>";
  $password ="wuhan";
  $st =sha1($password);
  if ($st ==='f2bd049d96fcfbd06ea9b8fc2130bf0be9fd1dd3'){
    echo "You are welcome.";
  }
?>
```

5. sha1_file()函数

sha1_file()函数用于计算文件的散列,该函数的语法格式为:

```
string sha1_file ( string $filename [, bool $raw_output = false ] )
```

利用美国安全散列算法 1，计算并返回由 filename 指定的文件的 SHA-1 散列值。该散列值是一个 40 字符长度的十六进制数字。

有关参数说明如下。

- $ filename：要散列的文件的名称。
- $ raw_output：如果被设置为 true，SHA-1 摘要将以 20 字符长度的原始格式返回。

返回值：成功则返回一个字符串，否则返回 false。

【示例 62】 使用 sha1_file()函数判断文件是否被修改。分为两个页面，eg62-1. php 用于求文件的散列，而 eg62-2. php 则用于判断文件两次散列是否相等。

eg62-1. php 代码如下。

```
<?php
  //加密文件 eg62.txt，得到字符串$sha1file
  $sha1file = sha1_file("eg62.txt");
  //文件加密后的字符串$sha1file 再写入文件 sha1file.txt 中
  file_put_contents("sha1file.txt",$sha1file);
?>
```

eg62-2. php 代码如下。

```
<?php
  //获得(eg62-1.php)加密后文件的内容并赋值给$sha1file，它是一个加密后的字符串
  $sha1file = file_get_contents("sha1file.txt");
  //再次加密原文件，得到字符串$st
  $st = sha1_file("eg62.txt");
  /* 将第一次加密文件的内容$sha1file 和第二次加密原文件的字符串$st 进行比较，若二者相
     等，说明文件没有被改变 */
  if ($st == $sha1file)
  {
    echo "The file is ok.";
  } else  {              //若二者不相等，说明文件被改变
    echo "The file has been changed.";
  }
?>
```

6. str_shuffle()函数

str_shuffle()函数用于随机打乱字符串中的全部字符，该函数的语法格式为：

```
string str_shuffle ( string $str )
```

str_shuffle() 函数打乱一个字符串，使用任何一种可能的排序方案。

说明：$ str 表示输入的字符串。

返回值：返回打乱后的字符串。

【示例 63】 使用 str_shuffle()函数随机打乱字符串。

eg63. php 代码如下。

```
<?php
  $str = 'abcdefghijklmnopqrstuvwxyz' ;
```

```
  $shuffled = str_shuffle($str);
  //输出类似于 dbcojvswuilgqeyfhaxtrkznpm
  echo $shuffled;
?>
```

7. base64_encode()函数

base64_encode()函数使用 MIME base64 对数据进行编码。该函数的语法格式为：

```
string base64_encode ( string $data )
```

使用 base64 对 data 进行编码。设计此种编码是为了使二进制数据可以通过非纯 8-bit 的传输层传输，例如电子邮件的主体。base64-encoded 数据要比原始数据多占用 33% 左右的空间。

说明：$ data 参数表示要编码的数据。

返回值：编码后的字符串数据，或者在失败时返回 false。

8. base64_decode()函数

base64_decode()函数用于对使用 MIME base64 编码的数据进行解码。该函数的语法格式为：

```
string base64_decode ( string $data [, bool $strict = false ] )
```

base64_decode()函数和 base64_encode()函数是互逆的。

有关参数说明如下。

- $ data：编码过的数据。
- $ strict：如果输入的数据超出了 base64 字母表，则返回 false。

返回值：返回原始数据，或者在失败时返回 false。返回的数据可能是二进制的。

【示例 64】 使用 base64_encode()函数对字符进行加密，然后使用 base64_decode()函数对加密的数据进行解密。

eg64.php 代码如下。

```
<?php
  echo "<pre>";
  $password = 'This is a password';           //待加密的数据
  $st = base64_encode($password);             //加密后的$st
  if ($st === 'VGhpcyBpcyBhIHBhc3N3b3Jk'){
     echo "You are welcome.\n";
  }
  echo base64_decode($st);                     //解密后
?>
```

7.4 正则表达式

正则表达式就是描述字符排列模式的一种自定义的语法规则，在 PHP 给我们提供的系统函数中，使用这种模式对字符串进行匹配、查找、替换及分隔等操作。它的应用非常广

泛。例如，常见的有：使用正则表达式去验证用户在表单中提交的用户名、密码、E-mail 地址、身份证号码及电话号码等格式是否合法；在用户发布文章时，将输入有 URL 的地方全部加上对应的链接；按所有标点符号计算文章中一共有多少个句子；抓取网页中某种格式的数据等。正则表达式并不是 PHP 自己的产物，在很多领域都会见到它的应用，除了在 Perl、C# 及 Java 语言中应用外，在我们的 B/S 架构软件开发中、Linux 操作系统、前端的 JavaScript 脚本、后台脚本 PHP 及 MySQL 数据库中都可以用到正则表达式。

7.4.1　正则表达式概述

正则表达式也称为模式表达式，自身具有一套非常完整的、可以编写模式的语法体系，提供了一种灵活且直观的字符串处理方法。正则表达式通过构建具有特定规则的模式，与输入的字符串信息比较中，在特定的函数中使用，从而实现字符串的匹配、查找、替换及分隔等操作。现在给出了三个模式，都是按照正则表达式的语法规则构建的，如下所示：

```
"/[a-zA-z]+://[^\s]*/" //匹配网址 URL 的正则表达式
"/<(\S*?)[^>]*>/*?</\1>|<.*?/>/i" //匹配 HTML 标记的正则表达式
"/\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w)*/" //匹配 E-mail 地址的正则表达式
```

我们不要被这些看似乱码的字符串给吓退，它们就是按照正则表达式的语法规则构建的模式，是一种由普通字符和具有特殊功能的字符组成的字符串，而且要将这些模式字符串放在特定的正则表达式函数中使用才有效果。

在 PHP 中支持两套正则表达式的处理函数库。一套是由 PCRE 库提供且与 Perl 语言兼容的正则表达式函数，使用以“preg_”为前缀命名的函数，而且表达式都应被包含在定界符中，如斜线(/)；另一套是由 POSIX 扩展语法的正则表达式函数，使用以“ereg_”为前缀命名的函数。两套函数库的功能相似，执行效率稍有不同。一般而言，实现相同的功能，使用第一种 PCRE 库提供的正则表达式效率略占优势，所以下面主要介绍使用“preg_”为前缀命名的正则表达式函数。

正则表达式由一些普通字符和一些元字符(metacharacters)组成。普通字符包括大小写的字母和数字，而元字符则具有特殊的含义，本书后面会给予解释。

在最简单的情况下，一个正则表达式看上去就是一个普通的查找串。例如，正则表达式"testing"中没有包含任何元字符，它可以匹配"testing"和"testing123"等字符串，但是不能匹配"Testing"。

要想真正地用好正则表达式，正确地理解元字符是最重要的事情。表 7-12(摘自《正则表达式之道》一书)列出了所有的元字符和对它们的简短描述。

表 7-12　特殊字符

元　字　符	说　　明
\	表示一个字符标记符，或表示一个向后引用，或表示一个八进制转义符。例如，“\\n”匹配\n，“\n”匹配换行符，序列“\\”匹配“\”，而“\(”则匹配“(”。即相当于多种编程语言中都有的“转义字符”的概念
^	匹配输入字符串的开始位置。如果设置了 RegExp 对象的 Multiline 属性，^也匹配“\n”或“\r”之后的位置

续表

元　字　符	说　　明
$	匹配输入字符串的结束位置。如果设置了 RegExp 对象的 Multiline 属性，$ 也匹配“\n”或“\r”之前的位置
*	匹配前面的子表达式任意次。例如，zo * 能匹配“z”，也能匹配“zo”以及“zoo”。* 等价于 o{0,}
+	匹配前面的子表达式一次或多次(大于或等于 1 次)。例如，“zo+”能匹配“zo”以及“zoo”，但不能匹配“z”；+等价于{1,}
?	匹配前面的子表达式零次或一次。例如，“do(es)?”可以匹配“do”或“does”中的“do”。“?”等价于{0,1}
{*n*}	*n* 是一个非负整数，匹配确定的 *n* 次。例如，“o{2}”不能匹配“Bob”中的“o”，但是能匹配“food”中的两个 o
{*n*,}	*n* 是一个非负整数，至少匹配 *n* 次。例如，“o{2,}”不能匹配“Bob”中的“o”，但能匹配“foooood”中的所有 o；“o{1,}”等价于“o+”；“o{0,}”则等价于“o * ”
{*n*,*m*}	*m* 和 *n* 均为非负整数，其中 $n \leqslant m$。最少匹配 *n* 次且最多匹配 *m* 次。例如，“o{1,3}”将匹配“fooooood”中的前三个 o 为一组，后三个 o 为一组。“o{0,1}”等价于“o?”。请注意在逗号和两个数之间不能有空格
?	当该字符紧跟在任何一个其他限制符(* 、+、?、{*n*}、{*n*,}、{*n*,*m*})后面时，匹配模式是非贪婪的。非贪婪模式尽可能少地匹配所搜索的字符串，而默认的贪婪模式则尽可能多地匹配所搜索的字符串。例如，对于字符串“oooo”，“o+”将尽可能多地匹配“o”，得到结果[“oooo”]，而“o+?”将尽可能少地匹配“o”，得到结果 ['o', 'o', 'o', 'o']
.(点)	匹配除“\r\n”之外的任何单个字符。要匹配包括“\r\n”在内的任何字符，请使用像“[\s\S]”的模式
(pattern)	匹配 pattern 并获取这一匹配。所获取的匹配可以从产生的 Matches 集合得到，在 VBScript 中使用 SubMatches 集合，在 JScript 中则使用 $0… $9 属性。要匹配圆括号字符，请使用“\(”或“\)”
(?:pattern)	匹配 pattern 但不获取匹配结果，不进行存储供以后使用。这在使用有“或”功能的字符"(|)"来组合一个模式的各个部分时很有用。例如“industr(?:y|ies)”就是一个比“industry|industries”更简略的表达式
(? =pattern)	非获取匹配，正向肯定预查，在任何匹配 pattern 的字符串开始处匹配并查找字符串，该匹配不需要获取供以后使用。例如，“Windows(? =95|98|NT|2000)”能匹配“Windows 2000”中的“Windows”，但不能匹配“Windows 3.1”中的“Windows”。预查不消耗字符，也就是说，在一个匹配发生后，在最后一次匹配之后立即开始下一次匹配的搜索，而不是从包含预查的字符之后开始
(?! pattern)	非获取匹配，正向否定预查，在任何不匹配 pattern 的字符串开始处匹配查找字符串，该匹配不需要获取供以后使用。例如“Windows(?! 95|98|NT|2000)”能匹配“Windows 3.1”中的“Windows”，但不能匹配“Windows 2000”中的“Windows”
(? <=pattern)	非获取匹配，反向肯定预查，与正向肯定预查类似，只是方向相反。例如，“(? <=95|98|NT|2000)Windows”能匹配“2000Windows”中的“Windows”，但不能匹配“3.1Windows”中的“Windows”
(? <! pattern)	非获取匹配，反向否定预查，与正向否定预查类似，只是方向相反。例如，“(?<!95|98|NT|2000)Windows”能匹配“3.1Windows”中的“Windows”，但不能匹配“2000Windows”中的“Windows”。 此处的任意一项都不能超过 2 位，如“(? <! 95|98|NT|20)Windows 正确，”(? <!95|98|NT|2000)Windows 报错。若是单独使用则无限制，如(? <!2000)Windows 可以正确匹配

续表

元　字　符	说　　明
$x\|y$	匹配 x 或 y。例如,“z|food”能匹配“z”或“food”(此处请谨慎)。“[z|f]ood”则匹配“zood”或“food”
[xyz]	字符集合。匹配所包含的任意一个字符。例如,“[abc]”可以匹配“plain”中的“a”
[^xyz]	负值字符集合。匹配未包含的任意字符。例如,“[^abc]”可以匹配“plain”中的“plin”
[a-z]	字符范围。匹配指定范围内的任意字符。例如,“[a-z]”可以匹配“a”到“z”范围内的任意小写字母字符。 **注意**:只有连字符在字符组内部时,并且出现在两个字符之间时,才能表示字符的范围;如果出字符组的开头,则只能表示连字符本身
[^a-z]	负值字符范围,匹配任何不在指定范围内的任意字符。例如,“[^a-z]”可以匹配任何不在“a”到“z”范围内的任意字符
\b	匹配一个单词边界,也就是指单词和空格间的位置(即正则表达式的“匹配”有两种概念,一种是匹配字符,一种是匹配位置,这里的\b 就是匹配位置的)。例如,“er\b”可以匹配“never”中的“er”,但不能匹配“verb”中的“er”
\B	匹配非单词边界。“er\B”能匹配“verb”中的“er”,但不能匹配“never”中的“er”
\cx	匹配由 x 指明的控制字符。例如,\cM 匹配一个 Control-M 或回车符。x 的值必须为 A～Z 或 a～z 之一。否则,将 c 视为一个原义的“c”字符
\d	匹配一个数字字符,等价于[0-9]
\D	匹配一个非数字字符,等价于[^0-9]
\f	匹配一个换页符,等价于\x0c 和\cL
\n	匹配一个换行符,等价于\x0a 和\cJ
\r	匹配一个回车符,等价于\x0d 和\cM
\s	匹配任何不可见字符,包括空格、制表符、换页符等,等价于[\f\n\r\t\v]
\S	匹配任何可见字符,等价于[^\f\n\r\t\v]
\t	匹配一个制表符,等价于\x09 和\cI
\v	匹配一个垂直制表符,等价于\x0b 和\cK
\w	匹配包括下画线的任何单词字符。类似但不等价于“[A-Za-z0-9_]”,这里的“单词”字符使用 Unicode 字符集
\W	匹配任何非单词字符,等价于“[^A-Za-z0-9_]”
\x*n*	匹配 *n*,其中 *n* 为十六进制转义值,十六进制转义值必须为确定的两个数字长。例如,“\x41”匹配“A”。“\x041”则等价于“\x04&1”。正则表达式中可以使用 ASCII 编码
\num	匹配 num,其中 num 是一个正整数,表示对所获取的匹配的引用。例如,“(.)\1”匹配两个连续的相同字符
n	标识一个八进制转义值或一个向后引用。如果*n* 之前至少有 *n* 个获取的子表达式,则 *n* 为向后引用。否则,如果 *n* 为八进制数字(0～7),则 *n* 为一个八进制转义值
nm	标识一个八进制转义值或一个向后引用。如果*nm* 之前至少有 *nm* 个获得子表达式,则 *nm* 为向后引用;如果*nm* 之前至少有 *n* 个获取,则 *n* 为一个后跟文字 *m* 的向后引用;如果前面的条件都不满足,若 *n* 和 *m* 均为八进制数字(0～7),则*nm* 将匹配八进制转义值 *nm*

续表

元　字　符	说　　明
nml	如果 *n* 为八进制数字(0～7),且 *m* 和 *l* 均为八进制数字(0～7),则匹配八进制转义值 *nml*
un	匹配 *n*,其中 *n* 是一个用 4 个十六进制数字表示的 Unicode 字符。例如,\u00A9 匹配版权符号“(©)”
\p{P}	小写 p 是 property 的意思,表示 Unicode 属性,用于 Unicode 正则表达式的前缀。中括号内的“P”表示 Unicode 字符集 7 个字符属性之一:标点字符。 其他 6 个属性如下(有的语言不支持)。 L:字母; M:标记符号(一般不会单独出现); Z:分隔符(比如空格、换行等); S:符号(比如数学符号、货币符号等); N:数字(比如阿拉伯数字、罗马数字等); C:其他字符
\\< \\>	匹配词(word)的开始(\\<)和结束(\\>)。例如,正则表达式\\<the\\>能够匹配字符串“for the wise”中的“the”,但是不能匹配字符串“otherwise”中的“the”。注意,这个元字符不是所有的软件都支持的
( )	将“(”和“)”之间的表达式定义为“组”(group),并且将匹配这个表达式的字符保存到一个临时区域(一个正则表达式中最多可以保存 9 个),它们可以用 \1 到\9 的符号来引用
|	将两个匹配条件进行逻辑“或”(Or)运算。例如正则表达式(him|her) 匹配“it belongs to him”和“it belongs to her”,但是不能匹配“it belongs to them.”。注意,这个元字符不是所有的软件都支持的

子字符类是嵌入较大字符类中的特殊字符类。除了自定义字符类(在其中定义要匹配的字符集,例如,[abxq4] 将匹配字符集限制为 a、b、x、q 和 4)以外,SQL Anywhere 还支持子字符类,比如大部分的 POSIX 字符类。[[:alpha:]] 表示所有大写和小写字母的集合。

REGEXP 搜索条件和 REGEXP_SUBSTR 函数支持表 7-13 中的所有语法约定,但 SIMILAR TO 搜索表达式不支持。SIMILAR TO 支持的约定在 SIMILAR TO 列中有一个 Y。

在 REGEXP 中,当使用 REGEXP_SUBSTR 函数时,可以使用脱字符对子字符类取非。例如,[[:^alpha:]] 匹配除字母字符以外的所有字符的集合。表 7-13 给出了子字符类及说明。

表 7-13　子字符类

子 字 符 类	说　　明
[:alpha:]	匹配当前归类中的大写和小写字母字符。例如,'[0-9]{3}[[:alpha:]]{2}' 匹配三个数字,后跟两个字母
[:alnum:]	匹配当前归类中的数字、大写和小写字母字符。例如,'[[:alnum:]]+' 匹配含有一个或多个字母和数字的字符串
[:digit:]	匹配当前归类中的数字。例如,'[[:digit:]-]+' 匹配含有一个或多个数字或横线的字符串。同样,'[^[:digit:]-]+' 匹配含有一个或多个不是数字或横线的字符的字符串

续表

子字符类	说　明
[:lower:]	匹配当前归类中的小写字母字符。例如，'[[:lower:]]' 不匹配 A，因为 A 为大写
[:space:]	匹配单个空格（' '）。例如，以下语句搜索 Contacts. City 以查找任何名称为两个词的城市： SELECT City FROM Contacts WHERE City REGEXP '. * [[:space:]]. * ';
[:upper:]	匹配当前归类中的大写字母字符。例如，'[[:upper:]ab]' 与以下其中一项匹配：任何大写字母、a 或 b
[:whitespace:]	匹配一个空白字符，例如，空格、制表符、换页符和回车符
[:ascii:]	匹配任何 7 位的 ASCII 字符(0～127 的顺序值)
[:blank:]	匹配一个空白区或水平制表符。[[:blank:]] 等效于 [\t]
[:cntrl:]	匹配顺序值小于 32 或字符值为 127 的 ASCII 字符(控制字符)。控制字符包括换行符、换页符、退格符，等等
[:graph:]	匹配打印字符。[[:graph:]] 等效于 [[:alnum:][:punct:]]
[:print:]	匹配打印字符和空格。[[:print:]] 等效于 [[:graph:][:whitespace:]]
[:punct:]	匹配其中一个字符： ! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [\ ] ^ _ ` { | } ~ . [:punct:] 子字符类不能包括当前归类中可用的非 ASCII 标点字符
[:word:]	匹配当前归类中的字母、数字或下画线字符。[[:word:]] 等效于 [[:alnum:]_]
[:xdigit:]	匹配字符类 [0-9A-Fa-f] 中的字符

7.4.2　Perl 风格正则表达式

正则表达式作为一个匹配的模板，是由原子(普通字符，例如 a～z)，有特殊功能的字符(元字符，例如 * 、+ 和? 等)，以及模式修正符 3 个部分组成。

在于 Perl 兼容的正则表达式函数中使用模式时一定要给模式加上定界符，即将模式包含在两个反斜线(/)之间。

1. 定界符

在使用 Perl 兼容的正则表达式时，要将模式表达式放入定界符之间。作为定界符不仅仅局限使用“/”，除了数字、字符和反斜线(\)以外的任何字符都可以作为定界符号。不过通常都习惯将模式表达式包含在两个斜线(/)之间。

2. 原子

原子是正则表达式的最基本组成单位，在每个模式中最少要包含一个原子。

(1) 普通字符作为原子

a～z、A～Z、0～9 等。

(2) 一些特殊字符和元字符作为原子

任何一个符号都可以当作原子使用，但是如果这个符号在正则表达式中表示一些特殊

含义,则必须使用转义字符取消其特殊含义,将其转变成普通原子。

(3) 一些非打印字符作为原子

所谓的非打印字符,就是在字符串中的格式控制符号,例如空格、回车符以及制表符。

(4) 使用“通用字符类型”作为原子

不管是打印字符还是非打印字符作为原子,都是一个原子只能匹配一个字符。而有时我们需要一个原子匹配一类字符,如匹配数字、匹配字母。

(5) 自定义原子表([])作为原子

可以自定义出特定的“类原子”,使用原子可以定义彼此平级的原子。

如“/[jhp]sp/”可以匹配 jsp、hsp、psp 三种。

3. 元字符

利用 Perl 正则表达式还可以使用各种元字符来搜索匹配。所谓元字符,就是用于构建正则表达式的具有特殊含义的字符,如“.、*、?、+”等。在一个正则表达式中,元字符不能单独出现,它是用来修饰原子的。如同我们中文的形容词一样,必须有形容的主体才可以。

构建正则表达式的方法和写文章的方法一样,就是使用多种形容词将某个人物或者事情清晰直观地表达出来。正则表达式的组件可以是单个的字符、字符集合、字符范围、字符间的选择或者是所有这些组件的任意组合。下面将这些字符分为几个小类来分别阐述。

(1) 限定符。限定次数。

(2) 边界限制。

(3) 原点。在字符之外,模式中的原点可以匹配目标中的任意一个字符,包括不可打印字符,但不匹配换行符。如果设定了模式修正符,则也会匹配换行符。

(4) 模式选择符(|)匹配选项中的任意一组。

(5) 模式单元。使用元字符()将多个原子组成大的原子,被当作一个单元独立使用。

(6) 后向引用。

(7) 模式匹配的优先级:从左到右。

7.5 正则表达式的使用方法

正则表达式只是作为对字符串格式进行匹配的表达式,其在程序中使用时需要借助正则表达式函数,可对字符串进行是否匹配的验证,并根据匹配的内容进行替换,或根据匹配字符串进行转义等。

7.5.1 正则表达式函数

PHP 为使用 Perl 兼容的正则表达式检索字符串提供了 7 个函数,这些函数具有不同的检索功能,如表 7-14 所示。

表 7-14　正则表达式函数

元　字　符	说　　明
preg_grep()	获取与模式匹配的数组单元
preg_match_all()	进行全局正则表达式匹配
preg_match()	进行正则表达式匹配
preg_quote()	转义正则表达式字符
preg_replace()	执行正则表达式的搜索和替换函数
preg_replace_callback()	通过回调函数执行正则表达式的搜索和替换
preg_split()	用正则表达式进行字符串分隔

从表 7-14 中可以看出,PHP 正则表达式不仅是用于检测字符串的格式是否匹配,而且还可以根据匹配的内容进行替换、分隔等操作。

7.5.2　正则表达式的匹配

正则表达式最基础的功能就是进行字符串的匹配。简单的匹配功能可以使用 preg_math()函数。该函数返回指定正则表达式对字符串的匹配次数,但是它的值只能是 0 次(不匹配)或 1 次(匹配),因为 preg_match()函数在第一次匹配后将停止搜索。preg_match()函数的语法格式为:

```
int preg_match ( string $pattern , string $subject [, array &$matches [, int $flags =0 [, int
$offset =0 ]]] )
```

有关参数说明如下。

- $pattern:要搜索的模式,为字符串形式。
- $subject:输入的字符串。
- $matches:如果提供了参数 matches,它将被填充为搜索结果。$matches[0] 将包含完整模式匹配到的文本,$matches[1] 将包含第一个捕获子组匹配到的文本。其他以此类推。
- $flags:$flags 可以省略,或被设置 PREG_OFFSET_CAPTURE 值。如果传递了这个标记,对于每一个出现的匹配,返回时会附加字符串偏移量(相对于目标字符串)。注意,这会改变填充到 matches 参数的数组,使其每个元素成为一个由第 0 个元素匹配到的字符串,第 1 个元素是该匹配字符串在目标字符串 subject 中的偏移量。
- $offset:通常搜索从目标字符串的起始位置开始。可选 offset 参数用于指定从目标字符串的某个位置开始搜索(单位是字节)。

返回值:preg_match()返回 $pattern 的匹配次数。它的值将是 0 次(不匹配)或 1 次,因为 preg_match()函数在第一次匹配后将会停止搜索。preg_match_all()函数不同于此,它会一直搜索 subject,直到到达结尾。如果发生错误,preg_match()函数返回 false。

【示例 65】 preg_match()函数举例。

```
<?php
  $str='1234509678';
```

```
    if (preg_match("/^[0-9]*(\.[0-9]+)*$/",$str)){
      echo '此字串由全数字组成';
    }
?>
```

程序运行结果如下。

```
此字串由全数字组成
```

7.5.3 正则表达式的全局匹配

同样是检测字符串与正则表达式是否匹配，preg_match_all()函数则会一直搜索到底，进行全局正则表达式的匹配，如果发生错误则返回 false。

preg_match_all()函数找出字符串中匹配模式的所有出现，通过可选的输入参数指定的顺序，将每次出现的匹配字符放在数组中。因此使用 preg_match_all()函数可以获取字符串中匹配的次数。preg_match_all()函数基本语法如下。

```
int preg_match_all ( string $pattern , string $subject [, array &$matches [, int $flags =
PREG_PATTERN_ORDER [, int $offset = 0 ]]] )
```

有关参数说明如下。

- $pattern：要搜索的模式，为字符串形式。
- $subject：输入的字符串。
- $matches：多维数组，作为输出参数输出所有匹配结果，数组排序通过 flags 指定。
- $flags：定义的排序方式。$flags 可以结合下面标记使用。
  - ✍PREG_PATTERN_ORDER：结果顺序为，$matches[0]保存完整模式的所有匹配，$matches[1] 保存第一个子组的所有匹配，以此类推。
  - ✍PREG_SET_ORDER：结果顺序为，$matches[0]包含第一次匹配得到的所有匹配(包含子组)，$matches[1] 包含第二次匹配到的所有匹配(包含子组)的数组，以此类推。
  - ✍PREG_OFFSET_CAPTURE：结果顺序为，$matches[0] 保存完整模式的所有匹配，$matches[1] 保存第一个子组的所有匹配，以此类推。
  - ✍PREG_SET_ORDER：结果顺序为 $matches[0] 包含第一次匹配得到的所有匹配(包含子组)，$matches[1] 包含第二次匹配到的所有匹配(包含子组)的数组，以此类推。
  - ✍PREG_OFFSET_CAPTURE：如果这个标记被传递，每个发现的匹配返回时会增加它相对目标字符串的偏移量。注意这会改变 matches 中的每一个匹配结果字符串元素，使其成为一个第 0 个元素为匹配结果的字符串，第 1 个元素为匹配结果字符串在 subject 中的偏移量。
- $offset：通常查找时从目标字符串的起始位置开始。可选 offset 参数用于从目标字符串中指定位置开始搜索(单位是字节)。

如果没有给定排序标记，则假定设置为 PREG_PATTERN_ORDER。注意，不能同时使用 PREG_PATTERN_ORDER 和 PREG_SET_ORDER。

返回值：返回完整匹配的次数(可能是 0)。如果发生错误则返回 false。

【示例 66】　preg_match_all()函数的应用举例。

```
<?php
  preg_match_all("|<[^>]+>(.*)</[^>]+>|U","<b>example:</b>
  <div align=\"left\">this is a test</div>",$out,PREG_SET_ORDER);
  echo $out[0][0].",".$out[0][1]."\n";
  echo $out[1][0].",".$out[1][1]."\n";
?>
```

程序运行结果如下。

```
example:,example:
this is a test
,this is a test
```

7.5.4　获取与模式匹配的数组单元

preg_grep()函数能够获取数组中与指定正则表达式相匹配的成员，返回由匹配成员组成的新的数组。

preg_prep()函数的语法格式如下。

```
array preg_grep ( string $pattern , array $input [, int $flags =0 ] )
```

返回给定数组 input 中与模式 pattern 匹配的元素组成的数组。

有关参数说明如下。

- $pattern：要搜索的模式，为字符串形式。
- $input：输入的数组。
- $flags：如果设置为 PREG_GREP_INVERT，则这个函数返回输入数组中与给定模式 $pattern 不匹配的元素组成的数组。

返回值：返回使用 $input 中 $key 作为索引的数组。

【示例 67】　preg_grep()函数的应用举例。

```
<?php
  $a =array('张飞','刘备','刘封','刘禅');
  $b =preg_grep("/^刘/",$a);
  var_dump($b);
?>
```

程序运行结果如下。

```
array(3) { [1]=>string(6) "刘备" [2]=>string(6) "刘封" [3]=>string(6) "刘禅" }
```

7.5.5　转义正则表达式字符

使用 preg_quote()函数能够将字符串中指定的字符转换成转义字符，其实际上就是给指定的字符添加反斜线。

preg_quote()函数用于返回某个字符串的 Perl 正则表达式，其语法形式为：

```
string preg_quote ( string $str [, string $delimiter =NULL ] )
```

preg_quote() 需要参数 $ str,并向其中每个正则表达式语法中的字符前增加一个反斜线。这通常用于有一些运行时字符串需要作为正则表达式进行匹配时。

相关参数说明如下。

- $ tr：输入的字符串。
- $ delimiter：如果指定了可选参数 delimiter,它也会被转义。这通常用于转义 PCRE 函数使用的分隔符。/是最通用的分隔符。

返回值：返回转义后的字符串。

preg_quote()函数需要参数 str 并向其中每个正则表达式语法中的字符前增加一个反斜线。这通常用于有一些运行时字符串需要作为正则表达式进行匹配时。

正则表达式特殊字符有：

```
. \ + * ?[ ^ ] $ ( ) { } = !<> | : -
```

【示例 68】 preg_quote()函数的应用举例。

```
<?php
  $keywords = '$ 40 for a g3/400';
  $keywords =preg_quote($keywords,'/');
  echo $keywords ;
  //输出 \$ 40 for a g3\/400
?>
```

7.5.6 正则表达式的搜索和替换函数

preg_replace()函数能够找出字符串中的匹配项并将其替换。preg_replace()函数的语法格式为：

```
mixed preg_replace ( mixed $pattern , mixed $replacement , mixed $subject [, int $limit =-1
[, int &$count ]] )
```

它执行一个正则表达式的搜索和替换。搜索 subject 中匹配 pattern 的部分,并以 replacement 进行替换。

相关参数说明如下。

- $ pattern：要搜索的模式。可以使用一个字符串或字符串数组。也可以使用一些 PCRE 修饰符,包括被弃用的'e'(PREG_REPLACE_EVAL)。
- $ replacement：表示替换的字符串或字符串数组。如果这个参数是一个字符串,并且 $ pattern 是一个数组,那么所有的模式都使用这个字符串进行替换;如果 $ pattern 和 $ replacement 都是数组,则每个 $ pattern 使用 $ replacement 中对应的元素进行替换;如果 $ replacement 中的元素比 $ pattern 中的少,则多出来的 $ pattern 使用空字符串进行替换。

$ replacement 中可以包含后向引用\\n 或(PHP 4.0.4 以上可用) $ n,语法上首选后者。每个这样的引用将被匹配到的第 n 个捕获子组捕获到的文本替换。n 可以是 0～99,\\0 和 $ 0 代表完整的模式匹配文本。捕获子组的序号计数方式为：代表捕获子组的左括

号从左到右，从 1 开始数。如果要在 $replacement 中使用反斜线，必须使用 4 个“\\\\”。（注：因为这首先是 PHP 的字符串，经过转义后是两个，再经过正则表达式引擎后才被认为是一个原始反斜线）。

当在替换模式下工作并且后向引用后面紧跟着另外一个数字（比如，在一个匹配模式后再接着增加一个原文数字），不能使用\\1 这样的语法来描述后向引用。比如，\\11 将会使 preg_replace()函数不能理解你希望的是一个\\1 后向引用紧跟一个原文 1，还是一个\\11 后向引用后面不跟任何东西。这种情况下的解决方案是使用\${1}1，这创建了一个独立的 $1 后向引用，一个独立的原文 1。

当使用 e 修饰符时，这个函数会转义一些字符（即 '、"、\ 和 NULL），然后进行后向引用替换。当这些完成后，请确保后向引用解析完后没有单引号或双引号引起的语法错误（比如，'strlen(\'$1\')+strlen("$2")'）。确保符合 PHP 的字符串语法，并且符合 eval 语法。因为在完成替换后，引擎会将结果字符串作为 PHP 代码并使用 eval 方式进行评估，再将返回值作为最终参与替换的字符串。

- $subject：要进行搜索和替换的字符串或字符串数组。如果 $subject 是一个数组，搜索和替换会在 $subject 的每一个元素上进行，并且返回值也会是一个数组。
- $limit：每个模式在每个 subject 上进行替换的最大次数，默认值是 -1（无限）。
- $count：如果指定该值，将会被填充为完成的替换次数。

返回值：如果 subject 是一个数组，则 preg_replace()函数返回一个数组，其他情况下返回一个字符串。如果匹配被查找到，替换后的 subject 被返回，其他情况下返回没有改变的 subject。如果发生错误，则返回 NULL。

【示例 69】 preg_replace()函数的应用举例。

eg69.php 代码如下。

```
<?php
  $string = 'The quick brown fox jumped over the lazy dog.';
  $patterns = array();
  $patterns[0] = '/quick/';
  $patterns[1] = '/brown/';
  $patterns[2] = '/fox/';
  $replacements = array();
  $replacements[2] = 'bear';
  $replacements[1] = 'black';
  $replacements[0] = 'slow';
  echo preg_replace($patterns, $replacements, $string);
?>
```

程序运行结果如下。

```
The bear black slow jumped over the lazy dog.
```

7.5.7　正则表达式的搜索和替换

preg_replace_callback()函数执行一个正则表达式搜索并且使用一个回调进行替换。基本语法格式为：

```
mixed preg_replace_callback ( mixed $pattern , callable $callback , mixed $subject [, int $limit =-1 [, int &$count ]] )
```

这个函数的行为除了可以指定一个 $ callback 替代 $ replacement 进行替换字符串的计算，其他方面等同于 preg_replace()。

有关参数说明如下。

- $ pattern：要搜索的模式，可以是字符串或一个字符串数组。
- $ callback：一个回调函数，在每次需要替换时调用，调用时函数得到的参数是从 subject 中匹配到的结果。回调函数返回真正参与替换的字符串。callback()函数仅用于 preg_replace_callback()函数一个地方的调用。在这种情况下，可以定义一个匿名函数作为 preg_replace_callback()函数调用时的回调，这样可以保留所有调用信息在同一个位置，并且不会因为一个不在任何其他地方使用的回调函数名称而占用函数名称空间。
- $ subject：要搜索替换的目标字符串或字符串数组。
- $ limit：表示每个 subject 字符串的最大可替换次数。默认是－1(无限制)。
- $ count：如果指定该变量，则表示将被填充为替换执行的次数。

返回值：如果 $ subject 是一个数组，preg_replace_callback()函数会返回一个数组，其他情况返回字符串。错误发生时返回 NULL。如果查找到了匹配内容，则返回替换后的目标字符串(或字符串数组)，其他情况 subject 将会无变化返回。

【示例 70】 preg_replace_callback()函数的应用举例。

eg70. php 代码如下。

```
<?php
  echo "<pre>";
  //将文本中的年份增加一年
  $text ="April fools day is 04/01/2002\n" ;
  $text ="Last christmas was 12/24/2001\n" ;
  //回调函数
  function next_year($matches)
  {
    //通常$matches[0]是完成的匹配,$matches[1]是第一个捕获子组的匹配,以此类推
     return $matches[1].($matches[2]+1);
  }
  echo preg_replace_callback("|(\d{2}/\d{2}/)(\d{4})|","next_year",$text );
?>
```

程序运行结果如下。

```
April fools day is 04/01/2003
Last christmas was 12/24/2002
```

7.5.8 使用正则表达式分隔字符串

preg_split()函数通过一个正则表达式分隔字符串。该函数的基本形式为：

```
array preg_split ( string $pattern , string $subject [, int $limit =-1 [, int $flags =0 ]] )
```

该函数的基本功能是通过一个正则表达式分隔给定字符串。相关参数说明如下。

- $pattern：用于搜索的模式，为字符串形式。
- $subject：输入的字符串。
- $limit：如果指定，将限制分隔得到的子字符串最多只有 limit 个，返回的最后一个子串将包含所有剩余部分。limit 值为－1、0 或 NULL 时都代表“不限制”。作为 PHP 的标准，可以使用 NULL 跳过对 flags 的设置。
- $flags：flags 可以是任何下面标记的组合(以位或运算 | 组合)：
  ✍PREG_SPLIT_NO_EMPTY。
  ✍PREG_SPLIT_DELIM_CAPTURE。
  ✍PREG_SPLIT_OFFSET_CAPTURE。

返回值：返回一个使用 $pattern 边界分隔 $subject 后得到的子字符串组成的数组。

【示例 71】 preg_split()函数的应用举例。

eg71.php 代码如下。

```
<?php
  //使用逗号或空格(包含" ",\r,\t,\n,\f)分隔短语
  $keywords=preg_split ( "/[\s,]+/" , "hypertext language, programming" );
  print_r ( $keywords );
?>
```

程序运行结果如下。

```
Array ( [0] =>hypertext [1] =>language [2] =>programming )
```

7.6　常用的 Web 验证

正则表达式不仅适用于字符串和字符串数组的处理，还可以在 Web 验证中被充分使用。如通过正则表达式来验证 E-mail 地址、邮政编码、电话号码等，常用的验证类型及其对应的正则表达式如表 7-15 所示。

表 7-15　Web 验证时常用的正则表达式

验 证 类 型	正则表达式
E-mail 地址	/^\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*/
URL	(((f|ht){1}(tp|tps)://)[-a-zA-Z0-9@:%_\+.~#?&//=]+)
邮政编码	/^[1-9]\d{5}$/
中文	/^[\x{4e00}-\x{9fa5}]+$/u
电话号码(086-区号-号码-分机)	^086-[1-9][0-9]{1,4}-[1-9][0-9]{4,7}-[0-9]{3,4}$
手机号码	/^1[34578]\d{9}$/
身份证号码	/^[1-9]\d{5}[1-9]\d{3}((0\d)|(1[0-2]))(([0|1|2]\d)|3[0-1])\d{3}(\d|x|X)$/

续表

验 证 类 型	正则表达式
IP 地址	/\b(?:(?:25[0-5]|2[0-4][0-9]|[01]? [0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]? [0-9][0-9]?)\b/

【示例 72】 Web 验证常用正则表达式的应用举例。

eg72. php 代码如下。

```
<?php
  $mobilephone="18000000000";
  if(preg_match("/^1[34578]\d{9}$/",$mobilephone)){
    echo '手机号码格式正确!<br />';
  }else{
    echo '手机号码格式不对!<br />';
  }
  $email = "abc@qq.com";
  if(preg_match("/^\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*/",$email)){
    echo '邮箱格式正确!<br />';
  }else{
    echo '邮箱格式不对!<br />';
  }
  $url = "http://www.163.com";
  if(preg_match("(((f|ht){1}(tp|tps)://)[-a-zA-Z0-9@:%_\+.~#?&//=]+)",$url)){
    echo '网址格式正确!<br />';
  }else{
    echo '网址格式不对!<br />';
  }
  $id = "42011119990929567x";
  if(preg_match("/^[1-9]\d{5}[1-9]\d{3}((0\d)|(1[0-2]))(([0|1|2]\d)|
  3[0-1])\d{3}(\d|x|X)$/",$id)){
    echo '身份证号码格式正确!<br />';
  }else{
    echo '身份证号码格式不对!<br />';
  }
  $ip = "127.0.0.1";
  if(preg_match("/\b(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]
  |2[0-4][0-9]|[01]?[0-9][0-9]?)\b/",$ip)){
    echo 'IP 地址格式正确!<br />';
  }else{
    echo 'IP 地址格式不对!<br />';
  }
?>
```

程序运行结果如下。

```
手机号码格式正确!
邮箱格式正确!
网址格式正确!
身份证号码格式正确!
IP 地址格式正确!
```

7.7　综合案例——考生信息处理

办公室交给信息化办公室一个电子表格文件 eg73.xls，该文件保存着某班级一次期中考试的成绩，该文件的内容如图 7-10 所示。现要求完成以下任务。

- 将考生信息保存在文本文件 eg73.txt 中。
- 将考生信息保存在数组中。
- 对考生信息进行排序，按照从高到低排序。
- 将排序后的考生信息输出在网页的表格中。

	A	B	C	D
1	学号	姓名	分数	性别
2	1	Liubei	79	male
3	2	Guanyu	75	male
4	3	Zhangfei	89	male
5	4	Diaochan	82	female
6	5	Machao	76	male
7	6	Lvbu	73	male
8	7	Caocao	99	male
9	8	Sunshangx	79	female
10	9	Dianwei	87	male
11	10	Sunce	56	male

图 7-10　考生成绩信息表

本节将用到字符串或者文件格式，还要用到数组和字符串处理函数。

步骤 1　为了便于 PHP 程序处理和不出现乱码，先将文件 eg73.xls 另存为 eg73.csv（以逗号分隔）的格式，并使用记事本打开，另存时的选项："编码"选择 UTF-8，"保存类型"选择"所有文件"，文件名为 eg73.txt（eg73.csv 格式也可以）。此时已经完成第一项任务：将考生信息保存在文本文件 eg73.txt 中。

步骤 2　为了将考生信息保存在数组中，先必须读入 eg73.txt 或者 eg73.csv 文件到字符串 $str 中；再分隔字符串为数组（$arr），即按照一行一行来分隔（按 Enter 键换行分隔），每一行是一个数组元素。代码如下。

```
header("content-type:text/html;charset=utf-8");  //用于网页编码，呼应步骤 1 中的编码
$file = 'eg73.csv';                              //或者为$file = 'eg73.txt';
$str = file_get_contents($file);                 //考生信息全部保存在字符串$str 中
$arr = explode("\r\n",$str);           //每个数组元素都保存一个考生信息（除首个元素为标题外）
```

注意： file_get_contents()函数用于获取文本文件的内容，以字符串形式返回。

步骤 3　再将每一行（$arr 数组的每一个元素）的字符串信息分隔为最基本的单元信息（用逗号来分隔）。至此，完成将考生信息保存到数组的操作。代码如下。

```
$stu_info = array();
foreach($arr as $value){
    $stu_info[] = explode(',',$value);
}                                      //至此，考生信息已保存到数组$stu_info 中
```

步骤 4　后面的处理与第 6 章中考生信息的处理类似。将考生的成绩信息保存到数组 $score 中。注意此处忽略考生信息的标题信息，因为标题不参与排序。代码如下。

```
$score = array();
for($i=1; $i<count($stu_info);$i++){
    $score[] = $stu_info[$i][2];
}                                      //取出数组中的成绩信息，为排序做准备
```

步骤 5 对考生成绩进行排序，排序需要保留键名。代码如下。

```
arsort($score);                        //按照成绩降序排列，保留键名
```

步骤 6 将考生信息 $stu_info 保存到另外一个数组 $stu_new 中，漏掉标题信息，且要排序。至此，考生信息排序已完成，代码如下。

```
$stu_new=array();
foreach($score as $k=>$v){
    $stu_new[]=$stu_info[$k+1];
}                                      //$stu_new 中保存着考生信息，不要标题
unset($stu_info);                      //释放掉原来的数组信息$stu_info
```

步骤 7 输出信息到网页表格中。至此，将考生信息输出到表格的操作已完成。代码如下。

```
echo "<table align='center' border='1' width='400'>";
echo "<tr>";
echo "<th>ID</th><th>name</th><th>score</th><th>sex</th>";
echo "</tr>";
foreach($stu_new as $value){
    echo "<tr>";
     foreach($value as $v){
       echo "<td>".$v."</td>";
     }
    echo "</tr>";
}
echo "</table>";                       //输出到表格
```

保存全部代码为 eg73.php。运行 eg73.php 程序，程序运行结果如图 7-11 所示。

ID	name	score	sex
7	Caocao	99	male
3	Zhangfei	89	male
9	Dianwei	87	male
4	Diaochan	82	female
8	Sunshangxiang	79	female
1	Liubei	79	male
5	Machao	76	male
2	Guanyu	75	male
6	Lvbu	73	male
10	Sunce	56	male

图 7-11 成绩处理后的输出结果

7.8　习　　题

一、填空题

1. 字符串连接使用________或“. =”符号。
2. 字符串可以使用单引号、双引号或________。
3. ________函数用于求子串。
4. chop()函数是________函数的别名。
5. 去掉字符串最右边指定的字符使用________函数。
6. 用正则表达式分隔字符串的函数是________函数。
7. 检查邮箱格式的正则表达式是________。
8. 检查电话号码格式的正则表达式是________。
9. 检查网址格式的正则表达式是________。

二、选择题

1. 下列函数将字符串的首字符转换为大写字符________。
   A. strouppe()　　B. strtrfirst()　　C. ucfirst()　　D. ucword()
2. 将字符串分隔为数组的函数是________。
   A. implode()　　B. explode()　　C. strpos　　D. stripos()
3. 将数组连接为字符串的函数是________。
   A. implode()　　B. explode()　　C. strpos　　D. stripos()
4. 从字符串中获取最右边 n 个字符的函数是________。
   A. implode()　　B. substr()　　C. strpos　　D. stripos()
5. 从字符串中获取最左边 n 个字符的函数是________。
   A. implode()　　B. substr()　　C. strpos　　D. stripos()
6. ________函数能够获取数组中与指定正则表达式相匹配的成员。
   A. preg_grep()　　B. preg_match()
   C. preg_replace()　　D. preg_replace_callback()
7. 使用________函数能够将字符串中指定的字符转换成转义字符。
   A. preg_quote()　　B. preg_grep()
   C. preg_replace()　　D. preg_replace_callback()

三、程序设计题

1. 下面是一段英文，执行以下一些操作。

Long long ago, there was a king. He liked to draw pictures. He thought his pictures were good, so he liked to show them to people. People were afraid to say that the king's pictures were bad, so they all said that his pictures were very good.

One day, the king showed some of his best pictures to an artist . He wanted the artist to speak well of these pictures. But the artist said his pictures were so bad that he should put them into the fire. The king got angry with him and put him to prison.

After some time, the king's guard brought the artist back to the palace. The king said to the artist. "I will set you free if you tell me which one of my pictures is good." Again he showed him some of his new pictures and asked what he thought of them.

After having a look at them, the artist at once turned to the guard and said, "Take me back to prison, please."

(1) 编写程序,获取第二段英文。

(2) 编写程序,统计整个字符串中"He"出现的次数。

(3) 编写程序,统计字符串中单词的个数。

2. 编写程序,要求验证电话号码、邮箱、网址和身份证号码的格式。

第 8 章　面向对象的程序设计

知识点：

- 面向对象的概念
- 类的定义、对象的创建
- 构造函数和析构函数
- 类的成员
- 抽象类
- 类的特性
- 接口的概念、定义和使用

本章导读：

PHP 支持面向对象的程序设计。面向对象程序设计(Object Oriented Programming，OOP)是一种程序设计范式，同时也是一种程序开发的方法。对象是指类的实例。它将对象作为程序的基本单元，将程序和数据封装其中，以提高软件的重用性、灵活性和扩展性。

本章将会引导我们进入面向对象的世界，学会使用面向对象的程序设计思想解决程序设计的实际问题。内容主要包括类的定义，对象的创建，构造函数、析构函数的使用，类的特性，接口的定义和使用，等等。

8.1　面向对象的编程

面向对象的编程起源于 20 世纪 60 年代，直到 20 世纪 90 年代才成为应用软件的开发主流，目前在网站开发中也盛行面向对象的编程。

8.1.1　理解面向对象编程

面向对象(Object Oriented，OO)是 PHP 的一个特点，利用面向对象的编程，对于提高 PHP 编程能力和规划好 Web 开发架构都非常有意义。面向对象编程有 3 个方法，分别是面向对象的分析(Object Oriented Analysis，OOA)、面向对象的设计(Object Oriented Design，OOD)和面向对象的编程。

面向对象的编程专注于软件对象的操作，是问题求解的一个途径，也是软件组织和组建的一种重要的方法。使用面向对象编程具备以下优点。

- 容易分发代码，方便系统的二次开发。
- 提高了代码的整洁性以及可重用性。
- 提高了程序的可扩展性。

- 适合于团队开发。
- 面向对象编程有很多设计模式可以利用,能够直接再次利用和开发。

8.1.2 面向对象编程的特性

面向对象编程有 3 个特性:封装、继承和多态。这 3 个特性也是 PHP 5 的特性。下面分别介绍这 3 个特性。

1. 封装

所谓封装,也就是把客观事物封装成抽象的类,并且类可以把自己的数据和方法只让可信的类或者对象操作,对不可信的进行信息隐藏。封装是面向对象的特征之一,是对象和类概念的主要特性。简单地说,一个类就是一个封装了数据以及操作这些数据的代码的逻辑实体。在一个对象内部,某些代码或某些数据可以是私有的,不能被外界访问。通过这种方式,对象对内部数据提供了不同级别的保护,以防止程序中无关的部分意外地改变或错误地使用了对象的私有部分。

2. 继承

所谓继承,是指可以让某个类型的对象获得另一个类型的对象属性的方法,它支持按级分类的概念。继承是指这样一种能力:它可以使用现有类的所有功能,并在无须重新编写原来类的情况下对这些功能进行扩展。通过继承创建的新类称为“子类”或“派生类”,被继承的类称为“基类”“父类”或“超类”。继承的过程,就是从一般到特殊的过程。要实现继承,可以通过“继承”(Inheritance)和“组合”(Composition)来实现。继承概念的实现方式有两类:实现继承与接口继承。实现继承是指直接使用基类的属性和方法而无须额外编码的能力;接口继承是指仅使用属性和方法的名称,但是子类必须提供实现的能力。

3. 多态

所谓多态,是指一个类实例的相同方法在不同情形下有不同的表现形式。多态机制使具有不同内部结构的对象可以共享相同的外部接口。这意味着,虽然针对不同对象的具体操作不同,但通过一个公共的类,它们(那些操作)可以通过相同的方式予以调用。

8.1.3 面向对象编程的原则

运用面向对象的思想进行软件设计时要遵循以下基本原则。

1. 单一职责原则

单一职责原则是指一个类的功能要单一,不能包罗万象。如同一个人一样,分配的工作不能太多,否则一天到晚虽然忙忙碌碌的,但效率却不高。

2. 开放封闭原则

开放封闭原则是指一个模块在扩展性方面应该是开放的,而在更改性方面应该是封闭的。比如,一个网络模块,原来只有服务端功能,而现在要加入客户端功能。

那么应当在不用修改服务端功能代码的前提下就能够增加客户端功能的实现代码,这要求在设计之初,就应当将服务端和客户端分开,将公共部分抽象出来。

3. 替换原则

替换原则是指子类应当可以替换父类并出现在父类能够出现的任何地方。比如,公司召开年度晚会,所有员工可以参加抽奖,那么不管是老员工还是新员工,也不管是总部员工

还是外派员工,都应当可以参加抽奖,否则该公司就不和谐了。

4. 依赖原则

依赖原则是指具体依赖于抽象,上层依赖于下层。假设 B 是比 A 低的模块,但 B 需要使用到 A 的功能,这个时候 B 不应当直接使用 A 中的具体类:而应当由 B 定义一个抽象接口,并由 A 来实现这个抽象接口,B 只使用这个抽象接口,这样就达到了依赖倒置的目的,B 也解除了对 A 的依赖;反过来是 A 依赖于 B 定义的抽象接口。通过上层模块难以避免依赖下层模块,假如 B 也直接依赖于 A 的实现,那么就可能造成循环依赖。一个常见的问题就是编译 A 模块时需要直接包含 B 模块的 CPP 文件,而编译 B 时同样要直接包含 A 的 CPP 文件。

5. 接口分离原则

接口分离原则是指模块间要通过抽象接口隔离开,而不是通过具体的类强耦合起来。

8.2　类和对象的概述

类是对象的抽象,而对象是类的具体实例。类是抽象的,不占用内存;而对象是具体的,占用存储空间。类是用于创建对象的蓝图,它是一个定义包括在特定类型的对象中的方法和变量的软件模板。

8.2.1　了解类和对象

我们在现实生活中可以接触到各种各样的实体,例如,猫、狗、星星、月亮、人等。描述人的有姓名、身高、体重、性别这些属性,此外描述人的还有吃饭、走路和睡觉。我们把这些特性(又称为属性)和行为具体定义为类。

类实际上是一个模板,它定义了一类事物的属性和行为(面向对象编程中又叫作方法或者函数),而在此基础上创建的特定实例就是对象。例如,人类是一个类,而人类的一个实例如张三就是一个对象。因此类是对象的概括、抽象;对象是类的实例,是具体的东西。

8.2.2　类的定义

在 PHP 中,使用 class 关键字来定义类,在该关键字的后面跟类名。再向后就是一对大括号,在大括号里面包含着类的定义。类的定义包括类的属性和类的方法(又称为函数)。其基本形式如下。

```
class 类名{
  //类的属性定义
  //类的方法定义
}
```

【示例 1】 类的定义。

eg1.php 代码如下。

```
<?php
  class student{
```

```
        private $name,$age;
        public $sex;
        function showinfo(){
            echo $this->name."<br />";
            echo $this->sex."<br />";
            echo $this->age."<br />";
        }
    }
?>
```

8.2.3 创建对象

在示例 1 中声明了一个简单的 PHP 类，如果要使用这个类，首先要创建这个类的对象，然后调用这个类的属性、方法或接口。实例化类的对象需要用到 new 关键字，其形式为：

```
$obj =new 类名([参数列表])
```

【示例 2】 类的使用。

eg2.php 代码如下。

```
<?php
    class student{
        public $name,$age,$sex;     //student 类的 3 个属性
        function showinfo(){        //student 类的一个函数
            echo $this->name."<br />";
            echo $this->sex."<br />";
            echo $this->age."<br />";
        }
    }

    $stu1 =new student();           //创建类的对象$stu1
    //调用类的属性
    $stu1->sex ='male';
    $stu1->name ='Liubei';
    $stu1->age =23;
    //调用类的函数(又叫方法)
    $stu1->showinfo();
?>
```

8.2.4 构造函数

当需要创建类的对象时，PHP 会在内存中创建和分配一定的内存区域给对象，供开发者调用。

例如“$stu1 = new student();”语句会自动提供一个无参数的构造函数供开发者调用。有时候这种默认的无参构造函数无法满足实际需要，例如示例 2 中创建对象 $stu1 之后，还要分别给对象的 3 个属性赋值，比较烦琐。幸好 PHP 支持用户显式地定义构造函数。如果定义了显式的构造函数，则默认的隐式构造函数不再存在。构造函数的基本形式为：

```
function __construct ([ arg1 [ , arg2 [ arg3 [ ...]]]]){
```

```
    //构造函数体
}
```

【示例 3】 构造函数的使用。

eg3. php 代码如下。

```
<?php
  class student{
    private $name,$age,$sex;          //student 类的 3 个属性
    function showinfo(){              //student 类的一个函数
      echo $this->name."<br />";
      echo $this->sex."<br />";
      echo $this->age."<br />";
    }
    function __construct($name,$age,$sex){
      echo "Hello,world!<br />";
      $this->name = $name;
      $this->age = $age;
      $this->sex = $sex;
    }
  }

  $stu1 = new student('Liubei','23','male');
  //创建类的对象$stu1,显示"Hello,world!",并给 3 个属性赋值
  $stu1->showinfo();                  //调用类的方法
?>
```

程序运行结果如图 8-1 所示。



图 8-1　构造函数的声明和调用

说明：在调用构造函数时，3 个参数'Liubei'、'23'、'male'按顺序分别传递给构造函数的 3 个形式参数 $ name、$ age、$ sex。而构造函数体中的 $ this->name 等语句中的 $ this 表示访问的是类本身的成员，也就是说 $ this->name 是类 student 中的属性 $ name。

程序中显式地定义了构造函数 function __construct($ name, $ age, $ sex)，默认的隐式的无参构造函数 function __construct()不再存在。

8.2.5　析构函数

析构函数与构造函数的功能刚好相反，它在一个对象销毁时调用：比如将对象赋值为空；比如用 unset 销毁一个对象，或者在 PHP 程序执行完成自动调用析构函数销毁对象、释放内存。析构函数有一个统一的命名，即__destruct()，若不显式地声明析构函数，则存在

一个默认的隐式析构函数。

【示例 4】 析构函数的使用。

eg4.php 代码如下。

```
<?php
  class student{
    private $name;                          //student 类的 1 个属性
    function __construct($name){
      $this->name =$name;
    }
    function showinfo(){
      echo $this->name."<br />";
    }

    function __destruct(){
      echo 'Good bye!<br />';
    }
  }
  $stu =new student('Zhangfei');          //调用构造函数创建对象$stu
  $stu1 =new student('Liubei');           //调用构造函数创建对象$stu1
  $stu1->showinfo();                      //调用 showinfo()函数来显示 liubei
  $stu2 =new student('Guanyu');           //调用构造函数来创建对象$stu2
  $stu2->showinfo();                      //调用 showinfo()函数来显示 Guanyu
  $stu2=NULL;                             //调用析构函数销毁$stu2,显示 Good bye
  unset($stu1);                           //调用析构函数销毁$stu1,显示 Good bye
  echo "ha ha<br />";
  /* 本语句执行完毕程序将结束,会自动调用析构函数销毁对象$stu,
    因此输出 ha ha 后,自动调用析构函数销毁$stu,显示 Goodbye */
?>
```

程序运行结果如图 8-2 所示。



图 8-2 析构函数的调用

8.3 类 的 成 员

类的成员可以有多种形式,例如常量、属性和方法。本节将分别介绍这些类的成员。

8.3.1　常量

常量在使用的过程中不能进行更改，在 PHP 中声明常量使用 const 关键字。

【示例 5】　类中的常量。

eg5.php 代码如下。

```
<?php
  class testconst{
    const PI =3.1416;
    const HOBBY ="sing";
    function testfun(){
      echo self::HOBBY."<br />";
      //类里面通过“self::”来访问类的常量
      echo self::PI."<br />";
    }
  }
  echo testconst::HOBBY."<br />";
  //通过类名直接访问类常量
  $t1 =new testconst();
  echo $t1->testfun();
  //“echo $t1->PI;” 错误，因为常量属于整个类，不属于某个具体对象
?>
```

说明：在类中可以通过“$ this->”或者“self::”访问自身的成员。静态变量和函数被访问时使用“self::”。在一个类的对象(实例)的上下文中使用其他方法和变量时用 this。

8.3.2　字段

PHP 中的字段用来描述类的某个方面的属性，实际上，它就是对象所具有的属性，用来表示实体的某种状态。通常在类的开始位置声明字段，并为字段赋初始值。

1. 字段的权限

在一个类中定义字段时，可以使用 public、private 和 protected 关键字来设置访问字段的权限。

- public：表示所修饰的字段是公有的，在任何地方都可以访问。在面向对象的程序中并不提倡使用公有字段，这样便于保护数据的安全。
- private：表示所修饰的字段为私有的，不能由实例化的对象去直接访问它，也不能继承给子类，只能在类中访问它，类外部的其他位置无法访问私有字段。
- protected：表示修饰的字段是受保护的。受保护的字段与私有字段有点相似，都可以在类的里面访问，但是还可以在继承的子类中访问。

2. 字段的调用

在类中声明字段与声明变量非常相似，不同的是字段在类中。在创建字段时可以直接为其赋值，字段的使用与变量的使用并不相同，它不直接使用，需要通过使用对象来调用。所以需要指定相应的对象来调用字段。基本形式如下。

```
Object->field_name;
```

【示例 6】 类中字段的访问。

eg6. php 代码如下。

```
<?php
  class person{
    public $xm='liubei';
    //公有字段在类的里面,子类、类对象都可以访问
    public static $name='liubei';
    //公有字段在类的里面,子类、类对象都可以访问
    private static $age=23;
    //私有字段只能在类的里面访问
    protected static $sex='male';
    //受保护的字段在类的里面和子类中都可以访问
    function testfun(){
      //只能是静态(static)的
      echo self::$name."<br />";
      echo self::$age."<br />";
      //私有字段,只能在类的里面访问
      echo self::$sex."<br />";
    }
  }
  class student extends person{
    //会从父类中继承$name 和$sex
    function showinfo(){
      echo self::$name."<br />";
      echo self::$sex."<br />";
    }
  }
  $p1 =new person();
  $p1->testfun();
  echo $p1->xm."<br />";
  //公有字段,在类的对象中也可以访问
  $s1 =new student;
  $s1->showinfo();
?>
```

程序运行结果如下。

```
liubei
23
male
liubei
liubei
male
```

8.3.3 属性

面向对象的程序设计中并不提倡使用公有字段,有些时候,可以在类中声明叫作属性的特色变量来完成公有字段的功能。在属性中可以获取或者设置字段的值,并在赋值或者取值时添加一些验证。

在 PHP 5 中并不提供属性处理功能,因此需要开发者自己实现这一功能。其主要实现方法有两种：第一种是使用__set()和__get()方法进行处理;第二种是使用 get×××()方法和 set×××()方法进行处理。

1. __set()和__get()方法

__set()方法用于给隐藏的字段赋值,并且还可以在为类的字段赋值之前添加一些验证类数据的代码。基本形式为：

```
boolean __set([string property_name] [, mix value_to_assign])
```

在上述形式中,__set()方法包含 property_name 和 value_to_assign 两个参数。其中第一个参数是必需的,表示要设置的属性的名称;第二个参数是可选的,表示要设置的属性的值。

__get()方法用于获取类的字段的值。使用__get()方法时,只需要向该方法传递要获取的字段的名称即可。基本形式为：

```
boolean __get([string property_name]);
```

【示例 7】 __set()方法和__get()方法的使用。

eg7.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  class person{
    public $sex;
    private $age,$name;
    function __get($property){
      echo "调用__get()函数获得私有属性的值<br />";
      if (isset($this->$property))
        return $this->$property;
      else
        return NULL;
    }

    function __set($property,$value){
      echo "调用__set()函数给私有属性赋值<br />";
      if (is_int($value)){
        if ($value<18 || $value>100)
          $this->$property=24;
        else
          $this->$property=$value;
      }else {
        $this->$property=$value;
      }
    }
  }
  $p1=new person();
  $p1->name='liubei';
  //本语句将调用__set()方法给私有属性 name 赋值为'liubei'
  echo $p1->name."<br />";
```

```
  //本语句将调用__get()方法获得私有属性 name 的值(值为'liubei')
  $p1->age=10;
  //本语句将调用__get()方法给私有属性 age 赋值,该语句具有一定的检验功能,最终得到值是 24,
    而不是 10
  echo $p1->age."<br />";
  //本语句调用__get()方法,获得私有属性 age 的值(值为 24)
  $p1->sex='male';
  //sex 是公有的,因此不会调用__set()方法
  echo $p1->sex;
  //sex 是公有的,因此不会调用__get()方法
?>
```

程序运行结果如图 8-3 所示。



图 8-3　__get()方法和__set()方法

说明:

- __get()方法在获得对象的私有属性的值时自动被调用。
- __set()方法在给对象的私有属性赋值时自动被调用,并具备一定的数据验证的功能。一个好的程序可以加一些数据验证功能,以提高程序的纠错功能。
- 本程序中若不定义__get()方法和__set()方法,则对象是无法访问(赋值或获取值)类的私有属性的。因此本程序若不定义__get()和__set()这两个方法,则程序会出错。
- 无论是__get()方法还是__set()方法,在处理公有字段时都不会被调用,虽然不建议在类中定义公有字段(或属性)。

2. set×××()和 get×××()方法

使用__set()方法和__get()方法可以处理一般性的问题。但是在处理比较复杂的面向对象的程序的属性时则无能为力。幸好在 PHP 中可以使用 set×××()方法和 get×××()方法来解决诸如此类的问题。

set×××()用于给某个具体的私有属性赋值,而 get×××()函数则用于获得某个具体的私有属性。这两种方法可以为专门的私有属性赋值或获取值,最大限度地做到个性化操作。

【示例 8】 set×××()方法和 get×××()方法的使用。

eg8.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  class person{
    private $age,$name;
```

```
    function setName($name){
      $this->name=$name;
    }
    function getName(){
      return $this->name;
    }
    function setAge($age){
      if ($age<18 || $age>100)
        $this->age=24;
      else
        $this->age=$age;
    }
    function getAge(){
      return $this->age;
    }
  }

  $p1=new person();
  $p1->setName('liubei');
  echo $p1->getName()."<br />";
  $p1->setAge(10);
  echo $p1->getAge();
?>
```

程序运行结果如图 8-4 所示。



图 8-4　set×××()方法和 get×××()方法的使用

8.3.4　方法

类中除了包含常量、字段和属性之外，还可以定义方法。当方法在类中被定义以后，就被称为类的方法或成员方法。其实类的方法和 PHP 中的函数是一样的，只是函数定义在类的外面，类的成员方法则定义在类的里面。此外在定义类的成员方法时，还需要为方法定义访问权限。访问权限包括三种类型：public、protected 和 private，默认是 public。在类中创建方法的基本形式为：

```
[public | protected | private] function method_name([参数列表]){
  //方法体
}
```

说明：

- 使用 public 定义的方法可以在对象中使用，可以在类的里面使用，还可以继承给子类。

- 使用 protected 定义的方法可以在类的里面使用,可以继承给子类,但是无法通过对象来访问。
- 使用 private 定义的方法只能在类的里面访问,不能在对象中访问,也不能继承给子类。

【示例 9】 类的方法。

eg9.php 代码如下。

```
<?php
  class person{
    private function sayhello(){
      echo "Hello,world!<br />";
    }
    public function showinfo(){
      self::sayhello();
      $this->sayhello();
      //sayhello()是私有方法只能在类的里面访问
      echo "Hello,every one!<br />";
    }
    protected function showme(){
      echo "My name is Ken!<br />";
    }
  }
  class student extends person{
    //从父类中继承了公有方法 showinfo()和受保护的方法 showme()
    function showfun(){
    //默认为公有方法
      $this->showme();
      //继承自父类中的受保护的方法
    }
  }

  $p1=new person();
  $p1->showinfo();
  //在对象中访问公有方法 showinfo()
  $s1=new student();
  $s1->showinfo();
  //在对象中访问公有方法 showinfo(),继承自父类
  $s1->showfun();
  //在对象中访问公有方法 showfun()
?>
```

8.3.5 静态成员

在前面读者已经了解到,在类中声明的常量、字段、属性和方法,在类被实例化为对象之后,可以访问类中允许访问的成员。在 PHP 中还提供了一种被称为静态成员的字段、方法或属性,无须将类实例化为对象即可直接访问。

在类中声明静态成员很方便,只需要在相应的静态成员之前加上关键字 static 即可。

【示例 10】　类的静态成员。

eg10.php 代码如下。

```
<?php
  class person{
    public static $name = 'liubei';
    public static $sex = 'male';
    public static function show(){
      echo self::$name."<br />";
    }
  }
  person::show();                     //静态方法可以通过“类名::方法”来访问
  echo person::$name.'<br />';        //静态属性可以通过“类名::属性”来访问
  $p1=new person();
  $p1->show();                        //静态方法可以通过“对象名->方法”来访问
  //echo $p1->name;                   //静态的属性不能通过“对象名->属性”访问
?>
```

说明：

- 在 PHP 中，在类内部可以通过“self::方法”来调用类自身的方法，在类的外面通过“类名::方法”来调用类的静态方法。另外，在类的子类中还可以通过“perent::父类中的静态方法”来调用父类中的静态方法。
- static 关键字用于声明静态成员，静态成员包括静态方法、静态字段和属性。对于静态成员可以这样来理解：
  - ✍一个静态成员会被类的所有对象共享，例如，在类的 A 对象中对静态成员的修改会影响到类的 B 对象。
  - ✍静态成员属于类，与对象无关。
  - ✍静态的属性不能通过“对象名->属性”来访问，静态属性可以通过“类名::属性”来访问。
  - ✍静态方法可以通过对象的实例来调用，即通过“对象名->方法”访问，还可以通过“类名::方法”来调用。

8.4　抽　象　类

抽象类(Abstract Classes)是一种不能被实例化的类，它用于为继承的子类定义界面，在抽象类中的成员方法可以有已经被实现的方法，也可以包括还没有被实现的成员方法。若某个类继承了抽象类，则该子类必须实现抽象类中全部的还没有实现的方法(即抽象方法)。

使用 abstract 关键字来声明抽象类，形式如下。

```
abstract class classname{
    //在此处声明一般的方法和抽象的方法以及定义属性、常量、字段等
}
```

抽象类和一般的类很相似，同样具有属性和方法，除了不被实例化外，在抽象类的定义

中一般都要包含抽象方法。抽象方法是不实现的(在抽象类的子类中实现),抽象方法前面要冠以关键字 abstract。抽象方法的形式如下。

```
abstract function function_name( [ arg1 [ , arg2 [ ...] ] ] ) ;
```

【示例 11】 抽象类和抽象方法。

eg11.php 代码如下。

```
<?php
  abstract class democlass{
    protected static $name,$age,$sex;
    function fun1(){
      echo "How are you?";
    }
    abstract function fun2();
    abstract function fun3($arg1,$arg2);
  }

  class sonclass extends democlass{
    //必须实现父类的全部抽象方法
    function fun2(){
    }
    function fun3($arg1,$arg2,$arg3=""){
    }
  }

  $son1 =new sonclass();   //创建抽象类的子类的对象
  $son1->fun1();           //子类从抽象类中继承了公有的一般方法 fun1()
  $son1->fun2();           //子类中实现了抽象类的抽象方法 fun2()
  $son1->fun3('','');      //子类中实现了抽象类的抽象方法 fun3(),第三个参数使用默认值
  $son1->fun3(4,5,6);      //子类中实现了抽象类的抽象方法 fun3()
?>
```

说明:

- 抽象方法使用关键字 abstract。抽象方法不实现,只声明(括号之外是分号,不带有一对大括号)。
- 包含抽象方法的类,一定要定义为抽象类,抽象类前面要冠以关键字 abstract;从语法上讲抽象类是可以不包含抽象方法的(毫无现实意义)。
- 抽象类中还可以包含一般方法、属性、字段、常量等成员。
- 抽象方法在子类中必须实现,并且一个抽象类被子类继承之后,抽象类中的全部抽象方法都必须被实现。

8.5 final 的使用

使用 final 声明的方法是最终方法,这种方法不能被重写;使用 final 声明的类是最终版,不能被继承。

【示例 12】 final 的使用(final 用于方法)。

eg12.php 代码如下。

```
<?php
  class person{
    final function testfinalfun(){
      echo "This is a final function!";
    }
  }

  class student extends person{
    function  testfinalfun(){
    //尝试改写父类中的最终方法 testfinalfun(),会出错!!!
      echo "This is a final function!";
    }
  }
?>
```

程序运行结果如下。

```
Fatal error: Cannot override final method person::testfinalfun() in D:\phpStudy\WWW\ch8\
eg12.php on line 13
```

意思是说,程序第 13 行出错,不能改写 person 类中的最终方法 testfinalfun()。

【示例 13】 final 的使用(final 用于类)。

eg13.php 代码如下。

```
<?php
  final class person {
    function  testfinalfun(){
      echo "This is a final function!";
    }
  }

  class student extends person{
  //继承最终类 person。出错!!!
    private $a,$b;
    function fun1(){
    }
  }
?>
```

程序运行结果如下。

```
Fatal error: Class student may not inherit from final class (person) in D:\phpStudy\WWW\ch8\
eg13.php on line 13
```

意思是说,程序第 13 行出错,student 类不能从最终类 person 继承。

8.6 实现类的特性

封装性、继承性和多态性是类的三个特性。下面将详细地讲解在 PHP 中是如何实现封装性、继承性和多态性的。

8.6.1 封装性

封装是将代码及其处理的数据捆绑在一起的一种编程机制，这样可以保证程序和数据不受外部干扰，不被误用。简言之，在类中将字段声明为私有的，然后再通过类的方法来设置或者获取字段的值，就是一个封装的过程。把属性、字段、常量和方法等类的成员封装起来，得到的结果就是类。类的封装性使用户无须关心类的功能是如何实现的，只需要进行调用即可。

【示例 14】 本例实现一个类，该类可以求三个数中的最大值或最小值，用户只需调用类的方法即可，无须知道类中是如何实现求最大值的具体过程的。

eg14. php 代码如下。

```
<?php
  class calculator{
    static function fun1($num1,$num2,$num3,$op){
      $max = $min = $num1;
      if ($num2 > $max) $max = $num2;
      if ($num3 > $max) $max = $num3;
      if ($num2 < $min) $min = $num2;
      if ($num3 < $min) $min = $num3;
      if ($op == 1)
        return $max;
      else
        return $min;
    }
  }

  echo calculator::fun1(4,5,6,1)."<br />";
  //求三个数中的最大值
  echo calculator::fun1(4,5,6,0)."<br />";
  //求三个数中的最小值
?>
```

说明：本程序用于求三个数中的最大值或最小值。当第三个数是 1 时，用于求三个数的最大值，否则就是求三个数的最小值。用户无须知道这个功能是如何实现的，只需知道 fun1()方法的 4 个参数的含义以及返回值含义即可。这就是典型的面向对象程序设计的封装性。

8.6.2 继承性

面向对象的程序设计的最大的特点是继承性。所谓继承性，就是子类从父类中获得类

的成员，实现类的复用，提高程序的利用效率和开发周期。继承而产生的新类叫作子类，被继承的类叫作父类、基类或超类。子类可以继承父类的部分或全部成员(包括构造方法)。一个父类可以被多个子类继承，一个子类只能从一个父类继承，即所谓的单继承。在 PHP 中通过关键字 extends 来实现类的继承。基本形式为：

```
class sonclass extends fatherclass{
    //类体
}
```

sonclass 就是子类，fatherclass 就是父类。父类中使用 public 或 protected 修饰的成员可以继承给子类，而使用 private 修饰的成员是不能继承给子类的。

【示例 15】 本例定义一个父类 person、一个子类 student，子类 student 从父类 person 继承成员。

eg15.php 代码如下。

```
<?php
  class person{
    public $name='Liubei';          //公有属性，会继承给子类
    private $age=23;                //私有属性，不会继承给子类
    protected $sex='male';          //受保护属性，会继承给子类
    function fun1(){                //公有方法，会继承给子类
      echo "1111<br />";
    }
    protected function fun2(){      //受保护方法，会继承给子类
      echo "2222<br />";
    }
    private function fun3(){        //私有方法，不会继承给子类
      echo "3333<br />";
    }
  }

  class student extends person{
   //从 person 中继承 $name、$sex、fun1()和 fun2()
   //$sex 和 fun2()仍然受保护，$name 和 fun1()仍然公有
   //$age 和 fun3()不会继承过来
   function fun4(){
     echo self::$name;
     echo self::$sex;
     self::fun1();
     self::fun2();
   }
  }
  $st =new student();
  $st->fun1();
  echo $st->name;
?>
```

程序运行结果如下。

```
1111
```

```
Liubei
```

说明：

- 父类中公有属性和方法会继承给子类，在子类中该属性和方法仍然是公有的，本例中父类 person 的公有属性 $name 会继承给子类 student，而且在子类 student 中该属性仍然是公有的；父类中的公有方法 fun1() 会继承给子类 student，在子类 student 中方法 fun1()仍然是公有方法。
- 父类中受保护的方法和属性会继承给子类，在子类中该受保护的属性和方法仍然是受保护的。本例中父类 person 的受保护属性 $sex 会继承给子类 student，而且在子类 student 中该属性仍然是受保护的；父类中受保护的方法 fun2()会继承给子类 student，在子类 student 中 fun2()仍然是受保护的方法。
- 父类中私有的属性 $age 不能继承给子类 student，父类中私有的方法 fun3()不能继承给子类 student。

此外，在子类中，还可以修改或者重写(覆盖)父类中同名的方法，以实现更加合适的业务逻辑。

【示例 16】 子类覆盖父类的属性、改写父类的方法。

eg16.php 代码如下。

```
<?php
  class person{
    public $name='Liubei';         //公有属性，会继承给子类
    private $age=23;               //私有属性，不会继承给子类
    protected $sex='male';         //受保护属性，会继承给子类
    function fun1(){               //公有方法，会继承给子类
      echo "1111<br />";
    }
    protected function fun2(){     //受保护方法，会继承给子类
      echo "2222<br />";
    }
    private function fun3(){       //私有方法，不会继承给子类
      echo "3333<br />";
    }
  }

  class student extends person{
   public $name = 'Guanyu';        //覆盖父类属性
   protected $sex = 'female';      //覆盖父类属性
   function fun1(){                //改写继承自父类的方法 fun1()
     echo "aaaa<br />";
   }
   function fun4(){
     echo self::$name;
     echo self::$sex;
     self::fun1();
     self::fun2();
   }
  }
```

```
  $st =new student();
  $st->fun1();
  echo $st->name;
?>
```

程序运行结果如下。

```
aaaa
Guanyu
```

说明：

- 子类从父类继承属性，但是子类还可以重新定义与父类同名的属性来覆盖父类中同名的属性。本例中在子类 student 中定义的属性 $name 和 $sex，它覆盖了父类 person 的同名属性 $name 和 $sex。
- 子类从父类继承方法，但是子类还可以重新定义与父类同名的方法来改写父类中同名的方法。本例在子类 student 中定义的方法 fun1()，它改写了父类 person 的同名方法 fun1()。

8.6.3　多态性

多态性是指在面向对象中能够根据使用类的上下文来重新定义或改变类的性质和行为。在 PHP 中不支持重载，但是可以通过如下方法来实现多态性，即一个对外接口，多个内部实现方法。也就是说同一个操作对于不同类的实例，将产生不同的执行结果。换言之，不同类的对象收到相同的消息时，将得到不同的结果。

【示例 17】　多态的实现。

eg17.php 代码如下。

```
<?php
  class Teacher{
    function drawChart($arg){
    //画图形，需要借助于另一个对象$arg
      $arg->draw();
    }
  }

  class Polygon{
  //本类画多边形
    function draw(){
      echo "draw a polygon.<br />";
    }
  }

  class Circle{
  //本类画圆
    function draw(){
        echo "draw a circle.<br />";
    }
  }
```

```
  $t1 =new Teacher();
  $p1 =new Polygon();
  $c1 =new Circle();
  $t1->drawChart($p1);          //画多边形
  $t1->drawChart($c1);          //画圆
?>
```

说明：

- 在其他语言里面可以将方法定义为多种格式(即重载)来实现多态性，但是 PHP 不支持重载。
- 本程序定义了 Teacher 类，类中定义了 drawChart() 方法用于画图形，但是 drawChart()方法画图形需要借助其他对象，例如 Polygon 类的对象和 Circle 类的对象，而且不同类的对象画出的图形是不一样的，这样即可实现类的多态性，有点类似于间接地实现了类似于 Java 中的重载。

8.7 接　口

我们已经知道，继承性只能实现类与类之间的单继承性，通过类的继承性可以实现父子关系的描述。但是如果想实现多继承性，就需要用到接口，接口提供了多重继承功能的实现。

8.7.1 接口概述

使用接口可以定义某种服务的一般规范，声明所需的函数和常量，但是不定义函数的实现，仅仅是一个函数和常量的声明。某个类实现了这个接口，就要实现该接口中的全部方法。

接口和抽象类都可以包含方法，但二者是有区别的，区别主要表现在以下两个方面。

- 定义不同：抽象类中可以包含一些方法，甚至是已经定义的方法，当然也可以是只声明不定义的方法；而接口只能包括方法的声明，不能定义(具体实现)。
- 用法不同：抽象类通过子类来继承，直接引用父类的方法并实现父类的方法；而接口是用来被实现的，实现一个接口就要实现(即定义)接口中的全部方法。

8.7.2 定义接口

使用接口可以指定某个类必须要实现的方法，接口中无须实现这些方法，即无须书写的具体内容。在 PHP 中使用关键字 interface 来定义接口。在接口中所有的方法都必须声明为 public。基本形式为：

```
interface interfacename{
  const namelist;
  function methodname([arg1[,arg2[,…]]]);
}
```

【示例 18】 定义接口。

eg18. php 代码如下。

```
<?php
  interface IDemo1{
    public function add($user);
    public function delete($id);
    public function modify($id,$user);
    public function display($id);
  }
?>
```

说明：

- 接口名一般以 I 为开头字母，以便区别于类。
- 接口中的方法声明为 public，而且必须是空的。
- 接口中的方法不能使用大括号，只能使用分号结束。
- 接口不需要添加任何修饰符。
- 接口中可以定义常量，使用方法同类，但是接口中的常量不能被子类或子接口覆盖。

8.7.3 实现接口

接口定义完成后，需要通过类来实现接口。实现接口的类通过 implments 关键字来实现接口中的全部方法。

【示例 19】 实现接口。

eg19. php 代码如下。

```
<?php
  interface IDemo1{
    public function add($user);                //以分号结尾，不能加大括号
    public function delete($id);               //以分号结尾，不能加大括号
    public function modify($id,$user);         //以分号结尾，不能加大括号
  }

  class testinterface implements IDemo1{      //实现接口中的全部方法
    public function add($user){
      echo "添加用户";
    }
    public function delete($id){
      echo "删除用户";
    }
    public function modify($id,$user){
      echo "修改用户";
    }
  }
?>
```

说明：testinterface 类实现了 IDemo1 接口中的全部方法。

【示例 20】 实现多个接口。

eg20. php 代码如下。

```
<?php
  interface IDemo1{
    public function add($user);
    public function delete($id);
    public function modify($id,$user);
  }

  interface IDemo2{
    public function modify($id,$user);
    public function display($id);
  }

  class testinterface implements IDemo1,IDemo2{     //实现两个接口中的全部方法
    public function add($user){
      echo "添加用户";
    }
    public function delete($id){
      echo "删除用户";
    }
    public function modify($id,$user){
      echo "修改用户";
    }
    public function display($id){
      echo "显示用户";
    }
  }
?>
```

说明：testinterface 类实现了 IDemo1 接口和 IDemo2 接口中的全部方法。

【示例 21】 子类实现接口并继承父类。

eg21. php 代码如下。

```
<?php
  interface IDemo1{
    public function add($user);
    public function delete($id);
    public function modify($id,$user);
  }

  interface IDemo2{
    public function modify($id,$user);
    public function display($id);
  }

  class father{
    function fun1(){
      echo "测试";
```

```
    }
  }
  class testinterface extends father implements IDemo1,IDemo2{
    public function add($user){
      echo "添加用户";
    }
    public function delete($id){
      echo "删除用户";
    }
    public function modify($id,$user){
      echo "修改用户";
    }
    public function display($id){
      echo "显示用户";
    }
  }
?>
```

说明：本例中，子类 testinterface 继承了父类 father，并实现了两个接口 IDemo1 和 IDemo2，也就是说实现了这两个接口中的全部方法。

8.8　综合案例——输出图形

本案例要求使用接口声明三个没有实现的方法分别用于打印正方向、长方形和三角形。然后再定义类来实现接口中的全部方法，最后要求定义对象并调用类中的方法实现图形的输出。

步骤 1　新建网页 eg22.php，并定义接口 print_shape，代码如下。

```
<?php
  interface print_shape{
    function print_square($n);                    //打印正方形
    function print_rectangle($rows,$cols);        //打印长方形
    function print_triangle($n);                  //打印三角形
  }
  //省略其他代码
?>
```

步骤 2　定义 Printgraph 类，用于实现 print_shape 接口中的 3 个方法，代码如下。

```
<?php
  //省略接口的定义代码
class Printgraph implements print_shape{
  function print_square($n){
        echo "下面打印正方形<br />";
        for($line =1 ; $line <=$n ; $line++){
        for($j =0 ;$j <$n; $j++)
           echo "* ";
        echo "<br />";
        }
  }
```

```
    function print_rectangle($rows,$cols){
      echo "下面打印长方形<br />";
      for($line =1 ; $line <=$rows ; $line++){
        for($j =0 ;$j <$cols; $j++)
          echo "* ";
        echo "<br />";
      }
    }

    function print_triangle($n){
      echo "下面打印三角形<br />";
      for($line =1 ; $line <=$n ; $line++){
        for($j =0;$j<$line;$j++)
          echo "* ";
        echo "<br />";
      }
    }
  //省略其他代码
?>
```

步骤 3　定义对象并调用方法来输出图形。

```
<?php
  //省略代码
  echo "<pre>";
  $shape =new Printgraph();
  $shape->print_square(5);
  $shape->print_rectangle(6,4);
  $shape->print_triangle(6);
?>
```

步骤 4　程序运行结果如图 8-5 所示。



图 8-5　打印图形程序输出的结果

8.9 习　　题

一、填空题

1. 封装、________和多态性是面向对象程序设计的 3 个特性。

2. 在 PHP 中，通过________操作符检测当前对象实例是否属于某一个类的类型，它返回布尔值。

3. 在 PHP 中，通过关键字________来定义抽象类或抽象方法。

4. 在 PHP 中定义析构函数时，需要通过________加入。

5. 将类的某一个方法定义为静态，可以使用关键字________。

6. 在接口中，方法只能定义为________，不能实现它。

7. 类实现某几个接口，就必须实现________。

8. 子类只能从________个父类中继承成员，但是却可以实现________个接口。

9. ________方法只声明不定义，包含________方法的类一定是________类。

10. 一个类中若没有定义构造函数，则一定存在一个隐含的________。

二、选择题

1. 在 PHP 中定义接口需要用到的关键字是________。

A. class　　B. final　　C. interface　　D. abstract

2. 使用 final 修饰的类________。

A. 可以继承　　B. 可以被覆盖

C. 既可以被继承又可以被覆盖　　D. 既不能被继承又不能被覆盖

3. 关于类的封装性、继承性和多态性，下面说法正确的是________。

A. PHP 的类不能实现封装性，但是可以实现继承性和多态性

B. 在 PHP 中实现了类的多继承性

C. 子类可以覆盖父类的成员

D. 子类可以覆盖父类的私有成员

4. 执行下面的程序，输出结果是________。

```
<?php
  class A{
    function __construct(){
  echo "Hello,world!";
}
    function showme(){
  echo "How are you.";
}
  }

  $a =new A();
?>
```

A. Hello，world!　　B. How are you.

C. Hello,world! How are you.　　　　D. 没有输出

5. 执行下面的代码,输出结果是________。

```
<?php
  class A{
    function __destruct(){
        echo "Good bye,world!";
    }
    function showme(){
        echo "How are you.";
    }
  }

  $a =new A();
  $a->showme();
?>
```

A. Good bye,world!　　　　B. How are you.

C. How are you. Good bye,world!　　　　D. 没有输出

6. 执行下面的代码,输出结果为________。

```
<?php
  class Father{
    function showme(){
        echo "How are you.";
    }
  }
  class Son extends Father{
    function showme(){
        echo "I'm fine.";
    }
  }
  $a =new Son();
  $a->showme();
?>
```

A. How are you.　　　　B. I'm fine.

C. How are you. I'm fine.　　　　D. 没有输出

7. 执行下面的代码,输出结果为________。

```
<?php
  class Father{
    function __construct(){
        echo "Hello world!";
    }
  }
  class Son extends Father{
    function __construct(){
        parent::__construct();
        echo "Hello world!!!!!";
    }
```

```
    }
    $a =new Son();
?>
```

A. Hello world!　　　　　　　　B. Hello world!!!!!

C. Hello world! Hello world!!!!!　　D. 没有输出

三、程序设计题

1. 编写程序,定义接口来实现两个数的加、减、乘、除,并编写类实现该接口。

2. 先定义类,然后在类中定义方法,用于判断一个数是否为素数,最后定义对象来判断某个数是否为素数。

3. 定义一个类,在类中定义两个方法,用于计算长方形的面积和周长,并定义对象测试。

第 9 章　PHP 表单应用

知识点：

- 表单的属性和使用方法
- 表单元素的属性以及表单元素的使用
- 表单的提交方法
- 表单的高级操作
- 使用表单进行用户注册

本章导读：

表单在网页中主要负责数据采集功能。一个表单有 3 个基本组成部分：①表单标签。这里面包含处理表单数据所用 CGI 程序的 URL 以及数据提交到服务器的方法。②表单域。包含文本框、密码框、隐藏域、多行文本框、复选框、单选框、下拉选择框和文件上传框等。③表单按钮。包括提交按钮、复位按钮和一般按钮，用于将数据传送到服务器上的 CGI 脚本或者取消输入，还可以用表单按钮来控制其他定义了处理脚本的处理工作。

9.1　表 单 概 述

表单是一个包含表单元素的区域。表单元素是允许用户输入信息的元素。在 PHP 中用户注册、登录、向程序中输入信息都要用到表单和表单元素。后台接收到表单数据要进行表单处理，并反馈给用户。

9.1.1　表单构成

要想掌握表单技术，首先得了解表单元素，也就是表单域，这些表单元素包含文本框、密码框、隐藏域、多行文本框、复选框、单选框、下拉选择框和文件上传框等。在 PHP 开发中用到的较为重要的表单元素如表 9-1 所示。

表 9-1　表单元素

表 单 元 素	说　　明
text	简单的文本框，input 的默认行为
password	密码框
search	搜索框
submit、reset、button	“提交”按钮、“重置”按钮、“普通”按钮
number range	只能输入数字的按钮，range 用于设置数字的范围
checkbox 、radio	复选框、单选框

续表

表 单 元 素	说　　明
image	生成一个图片按钮
color	生成颜色代码的按钮
email、tel、url	生成一个检测电子邮件、号码、网址的文本框
date、month、time、week、datetime、datetime-local	与日期和时间有关的表单元素
hidden	生成一个隐藏的控件
file	生成一个上传文件的组件

表单元素常见的属性如表 9-2 所示。

表 9-2　常见的表单属性

表 单 属 性	说　　明
name	表单元素的名称属性，PHP 可以通过名称属性获得表单元素的值
value	表单元素的值
type	表单元素的类型
id	表单元素的 id
size	规定输入字段的宽度
cols	多行文本框中显示时可以容纳的字符列数
rows	多行文本框中显示时可以容纳的字符高度(显示的行数)
selected	确定列表项是否被选中
checked	确定单选框、复选框是否被选中
required	规定必须在提交之前填写输入字段
placeholder	提供可描述输入字段预期值的提示信息(hint)

9.1.2　表单标记

在页面中添加表单元素时需要把表单元素放置在表单内部。表单用于声明表单元素。可以把表单理解为容器，用来存放表单元素。表单标记的语法为：

```
<form action=" url" method="GET | POST" name="name" id="id" target="target">
</form>
```

上述代码中的属性解释如下。

- action：指定一个处理提交表单的格式。它可以是一个 url 地址，提交给程序或者一个电子邮件地址。
- method：指定提交表单的 HTTP 方法。提交表单的方式有两种，因此 method 属性的值也是两个，即 POST 和 GET。
  ✍POST：POST 是向服务器传送数据，传送的数据量较大。
  ✍GET：GET 是从服务器上获取数据，传送的数据量较小。
- target：确定表单提交到哪里去。

9.1.3 按钮

表单中可以包含多种按钮，实现不同的功能。表单按钮用于用户在单击时执行相应的操作，但不同的按钮引发事件的条件和效果是不一样的。常用的表单按钮有“提交”按钮、“重置”按钮和一般按钮。

1. “提交”按钮

“提交”按钮用于把输入的内容提交到服务器。“提交”按钮的语法格式为：

```
<input type="submit" name="name" id="id" value="value">
```

有关属性说明如下。

- type：设置为 submit，表示它是“提交”按钮。
- name：按钮的名称属性。
- value：按钮上显示的文字。
- id：按钮的 id，可以用于设置外观样式，JavaScript 常常通过 id 访问表单元素（包括按钮）。

2. “重置”按钮

“重置”按钮用于重置表单。该按钮用于将表单中的数据清空，如可以将用户注册页面中输入的用户名、密码、姓名、年龄等各种信息清空。“重置”按钮的语法格式为：

```
<input type="reset" name="name" id="id" value="value">
```

有关属性说明如下。

- type：设置为 reset，表示它是“重置”按钮。
- name：按钮的名称属性。
- value：按钮上显示的文字。
- id：按钮的 id，用于设置外观样式，JavaScript 常常通过 id 访问表单元素（包括按钮）。

说明：“重置”按钮不需要编写代码，不需要添加事件脚本，单击该按钮即可将其所在表单中的表单元素清空。

3. 一般按钮

一般按钮所引发的事件需要用户自定义脚本，该按钮控制脚本处理，其语法格式为：

```
<input type="button" name="name" id="id" value="value" onclick="脚本">
```

有关属性说明如下。

- type：设置为 button，表示它是一般按钮。
- name：按钮的名称属性。
- value：按钮上显示的文字。
- id：按钮的 id，可以用于设置外观样式，JavaScript 常常通过 id 访问表单元素（包括按钮）。
- onclick：单击该按钮后，要去执行的脚本一般是客户端的 JavaScript 代码，不是 PHP 代码，因为一般按钮并不提交到后台服务器。

【示例 1】 通过“提交”按钮提交信息到服务器。

eg1. php 代码如下。

```
<form name="form1" method="post" action="">
  输入姓名<input type="text" name="text1" id="textfield" style="width:80px">
  <br /><!--换行-->
  <input type="submit" name="bt1" id="button" value="确定" >
</form>
<?php
  header("content-type:text/html;charset=utf-8");
  if (isset($_POST['bt1']))
    echo "Your name is:".$_POST['text1'];
?>
```

程序运行结果如图 9-1 所示，左边为单击“确定”按钮之前的效果，右边为单击“确定”按钮之后的效果。





图 9-1　提交前后效果比对

说明：程序中需要输入姓名，因此要用到文本框；需要提交姓名到服务器，因此需要“提交”按钮。无论“提交”按钮或文本框都需要放置在表单中，因此需要添加表单标记。

if (isset($_POST['bt1']))用于判断用户是否单击了“提交”按钮——“确定”按钮，该按钮的 name 属性是 bt1；“echo "Your name is:". $_POST['text1'];”语句中的 text1 是文本框的 name 属性，服务器获取表单元素的值都是通过 name 属性获取的。

<form name="form1" method="post" action="">和</form>是表单标记，全部表单元素需要放置在表单标记中间。method="post"表示表单的提交方式是 post。

表单元素属于 HTML 代码范畴，有的人喜欢写到 PHP 中，也是可以的，因此上面的代码可以写成：

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<form name='form1' method='post' action=''>";
  echo "输入姓名<input type='text' name='text1' id='textfield' style='width:80px'>";
  echo "<br />";
  echo "<input type='submit' name='bt1' id='button' value='确定' >";
  echo "</form>";
  if (isset($_POST['bt1']))
    echo "Your name is:".$_POST['text1'];
?>
```

【示例 2】 通过一般按钮判断文本框内容是否为空，若为空即给出提示信息。

eg2. php 代码如下。

```
<script language="javascript">
  function test(){
    if (document.getElementById('textfield').value=='')     //判断文本框是否为空
      alert('Please input your name!');
  }
</script>
<form name="form1" method="post" action="">
  Input name<input type="text" name="text1" id="textfield" style="width:80px">
  <br />
  <input type="button" name="bt1" id="button" value="test" onClick="test()" >
  <!--单击该按钮则调用 JavaScript 中的脚本 -->
</form>
```

程序运行结果如图 9-2 所示。



图 9-2　一般按钮调用 JavaScript 脚本

9.1.4　文本框

文本框是一种供用户提交数据的表单元素。文本框一般用来输入文字。比如用户注册或登录时需要输入用户名和密码。文本框的语法格式为：

```
<input type="text" name="name" value="" >
```

有关属性说明如下。

- type：值设置为 text，表示它是文本框，用于输入文本。
- name：文本框的名称属性，PHP 通过名称属性获取文本框的值(内容)。
- size：定义文本框的宽度，单位是字符宽度。
- maxlength：定义最多可以输入的字符数。
- value：文本框的值。

9.1.5　密码框

密码框用于输入密码。用法和文本框一样，只是隐藏输入的内容，以"•"的形式显示。但是其内容仍然是用户输入的内容。语法格式为：

```
<input type="password" name="name" value="value" >
```

除了 type 值为 password 外，其他用法与文本框一样。

【示例 3】　输入用户名和密码并提交到服务器。

eg3.php 代码如下。

```
<form name="form1" method="post" action="">
  输入用户名<input type="text" name="user"><br>
  输入密码<input type="password" name="mima"><br>
  <input type="submit" name="button1" id="button" value="确定">
</form>
<?php
  header("content-type:text/html;charset=utf-8");
  if (isset($_POST['button1'])){
    echo "你的姓名是:".$_POST['user']."<br />";
    echo "你的密码是:".$_POST['mima']."<br />";
  }
?>
```

程序运行结果如图 9-3 所示。左边是单击“确定”按钮前，右边是单击“确定”按钮后的结果。



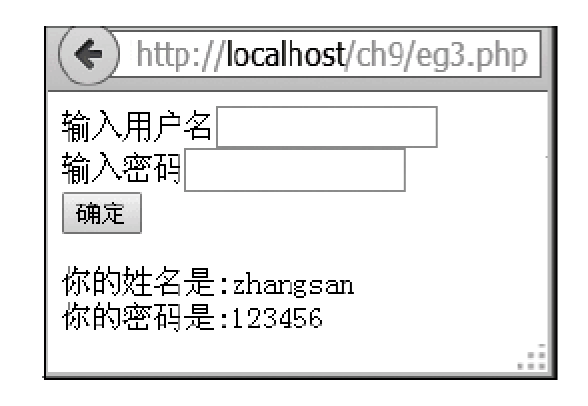


图 9-3　文本框和密码框程序运行结果

9.1.6　多行文本框

多行文本框支持文本的多行输入。它的语法格式为：

```
<textarea name="name" id="id" cols="cols" rows="rows" wrap="wrap">这里写内容
</textarea>
```

有关属性说明如下。

- cols：多行文本框的宽度，即列数。
- rows：多行文本框的高度，即行数。
- wrap：描述换行的属性，主要有如下几个值，即省略该属性；Off；Virtual；Physical。

【示例 4】　使用多行文本框输入备注信息。

eg4.php 代码如下。

```
<form name="form1" method="post" action="">
输入备注信息<br />
<textarea name="textarea1" id="textarea" cols="30" rows="5"></textarea><br />
<input type="submit" name="button1" id="button" value="提交">
</form>
<?php
  header("content-type:text/html;charset=utf-8");
```

```
  if (isset($_POST['button1'])){
    echo "备注信息是:".$_POST['textarea1'];
  }
?>
```

程序运行结果如图 9-4 所示，左边是输入内容但未提交的状态，右边是提交后的结果。



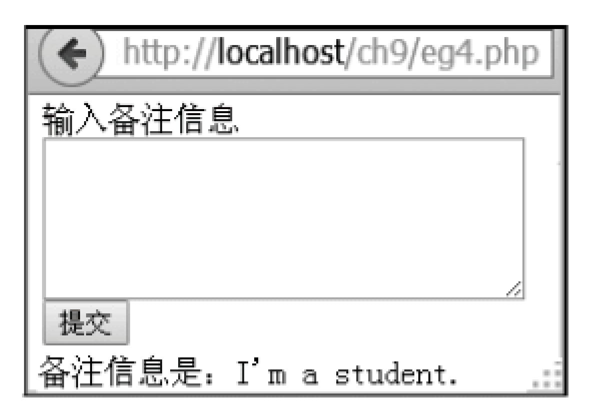


图 9-4　多行文本框

9.1.7　单选框

单选框列举多个选项供选择，但是每个选项之间是互斥的，只能选取其中一项或不选，例如性别只能选择男或女。单选框语法格式为：

```
<input type="radio" name="name" value="value" checked="checked">
```

有关属性说明如下。

- type：值为 radio，表示单选框。
- name：名称属性，可以用于分组。
- value：选中该项之后得到的值。
- checked：设置选中状态。

【示例 5】 使用单选框输入性别和班级。

eg5.php 代码如下。

```
<form name="form1" method="post" action="">
性别
<input type="radio" name="sex" value="male" checked >男<!--默认选中-->
<input type="radio" name="sex" value="female">女<br />
班级
<input type="radio" name="class" value="一班">一班
<input type="radio" name="class" value="二班">二班<br />
<input type="submit" name="bt1" value="确定" />
</form>
<?php
  header("content-type:text/html;charset=utf-8");
  if (isset($_POST['bt1'])){
     echo "性别是:".$_POST['sex']."<br />";
     echo "班级是:".$_POST['class'];
  }
?>
```

程序运行结果如图 9-5 所示，左边为选择性别和班级但未提交，右边是提交后的效果。



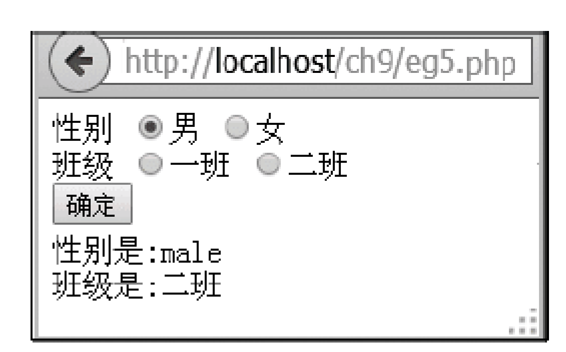


图 9-5　单选框的使用

本例中，性别的 name 都是 sex，因此男和女是一组；二者的 value 分别是 male 和 female，表示选中后返回的值。班级的 name 属性为 class，因此两个选项是一组，value 分别是“一班”和“二班”，是选中后返回的班级值。提交之后单选框的选中状态与所选项无关。

9.1.8　多选框

多选框用于多选多，有时需要根据多个选项来选择其中多个，这时可以使用多选框。多选框的多个选项之间互相不影响，可以全部选中，也可以一个都不选中，各选项是独立的。多选框语法格式为：

```
<input type="checkbox" name="name" value="value" checked>
```

有关属性说明如下。

- type：设置为 checkbox，表示它是复选框。
- name：名称属性，与分组有关，一般用于数组操作。
- value：选中本项后的返回值。
- checked：初始状态为选中。

【示例 6】 使用多选框输入爱好。

eg6.php 代码如下。

```
<form name="form1" method="post" action="">
体育爱好
<input type="checkbox" name="sport[]" value="足球" >足球   
<input type="checkbox" name="sport[]" value="游泳" checked>游泳   
<!--默认选中“游泳”选项-->
<input type="checkbox" name="sport[]" value="短跑" >短跑<br />
文艺爱好
<input type="checkbox" name="art[]" value="唱歌" >唱歌   
<input type="checkbox" name="art[]" value="跳舞" >跳舞   
<input type="checkbox" name="art[]" value="弹琴" >弹琴<br />
<input type="submit" name="button1" value="提交" />
</form>
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  if (isset($_POST['button1'])){
    //体育爱好和文艺爱好都是数组，使用 implode 来将数组整合为字符串
    echo "体育爱好有:".implode(",",$_POST['sport'])."\n";
    echo "文艺爱好有:".implode(",",$_POST['art']);
  }
?>
```

程序运行结果如图 9-6 所示，左边选择未提交，右边是提交后的结果。





图 9-6 使用复选框选择爱好

说明：为了方便程序批量处理，将体育爱好中的三个复选框设置为一组，其 name 属性设置为数组“sport[]”，而文艺爱好的三个复选框也设置为一组，其 name 属性为“art[]”。在服务器处理多选框时，分别得到 $_POST['sport']和 $_POST['art']两个数组，再使用 implode 获得数组的每个元素，使用逗号连接成字符串。

9.1.9 下拉列表框

下拉列表框与文本框类似，但是右边多一个向下的箭头，可将要选择的列表项展开，用户可以在多个列表项中进行选择。下拉列表框的语法格式为：

```
<select name="name" size="size" multiple>
<option value="value1" [selected]>tille1</option>
<option value="value2" [selected]>tille2</option>
...
<option value="value2" [selected]>tille2</option>
</select>
```

有关属性说明如下。

- size：定义下拉列表的行数。
- name：下列列表的名称。
- multiple：设置此项就可以多选，否则只能单选。
- value：定义选中项的值。
- selected：表示默认已经选择本选项。

【示例 7】 使用下拉列表框选择职业，职业只能有一项。

eg7. php 代码如下。

```
<form name="form1" method="post" action="">
  <select name="select1" id="select" >
    <option value="" selected>请选择职业</option><!--默认选项-->
    <option value="doctor">医生</option>
    <option value="teacher">教师</option>
    <option value="sproter">运动员</option>
    <option value="worker">工人</option>
  </select>  
  <input type="submit" name="bt1" value="确定" />
</form>
<?php
  header("content-type:text/html;charset=utf-8");
```

```
  if (isset($_POST['bt1'])){                        //单击了"提交"按钮——"确认"按钮
    if (strlen($_POST['select1'])==0)               //若选择了第 1 项
      echo "<script>alert('请选择职业')</script>";//若选择第 1 项,则弹出提示对话框
    else
      echo "你的职业是:".$_POST['select1'];         //若选择第 2~5 中的某项,则输出职业
  }
?>
```

说明：本程序中，第 1 项 value 值设置为空字符串，当选中该项时返回空字符串，可以作为未选择职业的标记。第 1 项中有 selected，表示默认选中该项。每一项都有 value，若选中该项，则返回该 value 值，而<option >和</option>之间的文字则是选项的标题。

【示例 8】 使用下拉列表框选择爱好，爱好可以有多项。

eg8. php 代码如下。

```
<form name="form1" method="post" action="">
  <select name="select1[]" id="select" multiple="multiple" >
    <option value="football">足球</option>
    <option value="basketball">篮球</option>
    <option value="vollyball">排球</option>
    <option value="ping-pong ball">乒乓球</option>
  </select>  
  <input type="submit" name="bt1" value="确定" />
</form>
<?php
  header("content-type:text/html;charset=utf-8");
  if (isset($_POST['bt1'])){
    echo "你的爱好是:".@implode(',',$_POST['select1']);
  }
?>
```

程序运行结果如图 9-7 所示。左边为提交前结果，右边为提交后的结果。





图 9-7　使用列表框多选

说明：通过设置列表框 select1 的 name 属性为 name="select1[]"，用数组来获取多个选项值。然后再通过 implode()函数将多个选项(数组)连接在一起。

9.1.10　文件上传框

文件上传框负责把文件上传到服务器。文件上传框类似文本框，但是比文本框多了一个文件浏览按钮。访问者可以通过输入需要上传的文件路径或者单击"浏览"按钮选择需要上传的文件。在使用文件上传框时需要注意以下几点。

（1）在文件上传时要确保服务器允许文件上传。

（2）表单标签中要设置 ENCTYPE="multipart/form-data"来确保文件正确编码。

（3）表单的传送方式必须是 POST。

文件上传框语法格式为：

```
<input type='file' name='name' size='size' maxlength='maxlength'>
```

有关属性说明如下。

- type：必须设置为 file，表示文件上传框。
- name：定义文件上传框的名称属性。
- size：定义文件上传框的宽度，单位是单个字符宽度。
- maxlength：定义最多输入的字符数。

要想获得上传文件的相关信息，可以使用 $_FILES['name']来获得。$_FILES['name']是一个关联数组。该数组共有 5 个元素，name 表示上传文件的文件名，type 表示上传文件的类型；tmp_name 是上传文件的临时位置和文件名；error 表示上传过程中的错误信息；size 表示上传文件的大小。有关知识点以及完整的文件上传示例将在文件相关章节详细介绍。

【示例 9】 使用文件上传框。

eg9.php 代码如下。

```
<form action="" method="post" enctype="multipart/form-data" name="form1">
  <input type="file" name="fileField1" id="fileField">
  <input type="submit" name="bt1" id="button" value="提交">
</form>
<?php
  header("content-type:text/html;charset=utf-8");
  echo '<pre>';
  if (isset($_POST['bt1'])){
     var_dump($_FILES['fileField1']);
  }
?>
```

程序运行结果如图 9-8 所示。左边是没有选择文件的提交，右边是选择文件再提交。

```
选择文件 未选择任何文件 提交

array(5) {
  ["name"]=>
  string(0) ""
  ["type"]=>
  string(0) ""
  ["tmp_name"]=>
  string(0) ""
  ["error"]=>
  int(4)
  ["size"]=>
  int(0)
}
```

```
选择文件 未选择任何文件 提交

array(5) {
  ["name"]=>
  string(10) "论文.txt"
  ["type"]=>
  string(10) "text/plain"
  ["tmp_name"]=>
  string(60) "C:\Documents and Setting..."
  ["error"]=>
  int(0)
  ["size"]=>
  int(589)
}
```

图 9-8 文件上传框

说明：图中右边 $_FILES['fileField1']['name']="论文.txt"表示上传的文件是“论文.txt”。

图中右边 $_FILES['fileField1'][type]= "text/plain"表示上传的文件类型是纯文本文件。

图中右边 $_FILES['fileField1'][tmp_name]= "C:\Documents and Settings\hdh\Local Settings\Temp\php257.tmp"表示上传的文件保存在服务器上的临时文件名。

图中右边 $_FILES['fileField1']['error']=0 表示上传的文件无错误。

图中右边 $_FILES['fileField1']['size']=598 表示上传文件的大小。

9.1.11　邮箱输入框

邮箱输入框用于输入邮件地址，在表单提交时会自动验证邮件地址(格式)是否正确。邮箱输入框的语法格式为：

```
<input type="email" name="name" multiple required .../>
```

有关属性说明如下。

- type：设置为 email，表示它是邮件输入框。
- name：属性，邮件输入框的名称属性。
- multiple：可以输入多个邮件，用逗号隔开。
- required：表示不能空着，必须有邮件输入。

【示例 10】　使用邮件输入框。

eg10.php 代码如下。

```
<form name="form1" method="post" action="">
  <input type="email" name="email1" multiple="multiple" style="width:180px" required/>
  <!--required 表示若不输入直接提交则提示"请填写此字段"-->
  <!--style="width:180px"用于设置邮件输入框宽度-->
  <!--multiple="multiple"表示可以输入多个邮件地址,用逗号隔开-->
  <input type="submit" name="bt1" value="确定" />
</form>
<?php
  header("content-type:text/html;charset=utf-8");
  if (isset($_POST['bt1'])){
    echo $_POST['email1'];
  }
?>
```

程序运行结果如图 9-9 所示。左边没有输入邮箱地址，右边是输入邮箱地址并单击"确定"按钮后的程序运行结果。



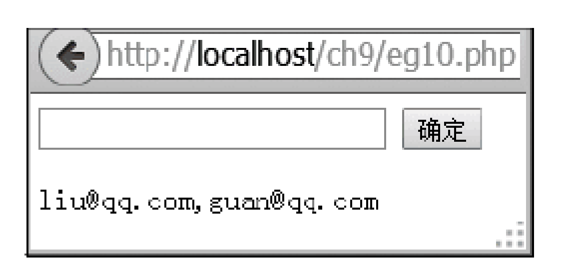


图 9-9　邮件输入框

9.1.12　电话输入框

如果需要一个用来填写电话号码的输入框，可以使用 tel 类型。这个在移动浏览器中使用很方便，会自动使用只包含数字和符号而没有字母的虚拟键盘。电话输入框的语法格式为：

```
<input type="tel" name="name" .../>
```

有关属性说明如下。

- type：设置为 tel，表示它是电话输入框。
- name：电话输入框的名称属性。

【示例 11】 使用电话输入框。

eg11.php 代码如下。

```
<form name="form1" method="post" action="">
  <input type="tel" name="tel1" required/>
  <input type="submit" name="bt1" value="确定" />
</form>
<?php
  header("content-type:text/html;charset=utf-8");
  if (isset($_POST['bt1'])){
    echo $_POST['tel1'];
  }
?>
```

9.2　表 单 提 交

表单提交在 PHP 中很重要，PHP 运行在服务器上，而用户输入表单中的内容在浏览器端，这需要提交到服务器端并由 PHP 程序处理。

9.2.1　表单的提交方式

通过设置表单的 method 属性来设置表单的提交方式。表单的提交方式有两种，分别是 POST 方式和 GET 方式。二者主要有以下区别。

（1）GET 是表单默认的提交方式，用于从服务器获取数据，而 POST 用于向服务器上传数据。

（2）GET 是将表单中的数据按照“参数名＝参数值”的形式添加到 action 所指向的 URL 后面，并且二者使用？连接，而各个变量之间使用 & 连接。POST 是将表单中的数据放在表单体中，按照“变量/值”相应的方式传递到 action 指定的 URL。

（3）GET 不安全，因为传输的数据放置在 URL 中。而现在诸多服务器、代理服务器或用户代理都会请求将 URL 记录到日志文件中，这样有可能将一些信息暴露给第三方。此外，一些系统内部的消息同时会显示在用户面前。而 POST 提交方式则不存在上述问题，因此 POST 提交方式是相对比较安全的。

(4) GET 传输的数据量比较小,POST 传输的数据量比较大,所以上传文件时只能使用 POST 提交方式。

(5) GET 显示表单数据值必须采用 ASCII 字符,而 POST 则没有这个限制。

9.2.2　表单的 GET 提交方式

在 PHP 页面中使用 $_GET 变量获取通过 GET 方式提交的数据,该变量是一个数组,数组的键是表单元素的名称属性。例如,表单中有一个 name 属性为 text1 的文本框,获取该文本框的内容可以使用 $_GET['text1']。此外还要设置表单标签的 method 属性为 get。使用 GET 方式提交的信息或显示在地址栏中,而且提交的信息量也是有限度的。

【示例 12】　表单的 GET 提交方式。

eg12.php 代码如下。

```
<form name="form1" method="get" action="">
<input type="text" name="t1" /><br />
<input type="password" name="p1" /><br />
<input type="submit" name="bt1" value="ok" />
</form>
<?php
  if (isset($_GET['bt1'])){
    echo $_GET['t1']."<br />";
    echo $_GET['p1'];
  }
?>
```

程序运行结果如图 9-10 所示。图 9-10 是在文本框中输入 ab,密码框中输入 12,然后单击 OK 按钮的程序运行结果。



图 9-10　使用 GET 方式提交表单

说明:表单标签中的 method="get"表示表单的提交方式是 get。

提交后网址显示为 http://localhost/ch9/eg12.php? t1=ab&p1=12&bt1=ok。其含义为:t1=ab,t1 是文本框的 name 属性,ab 是文本框的内容;p1=12,p1 是密码框的 name 属性,12 是密码框的内容;bt1=ok,bt1 是"提交"按钮的 name 属性,ok 是"提交"按钮显示的标题。

不同的表单元素之间使用 & 连接;表单的名称和值之间使用=连接;传递的数据与网址之间使用? 连接。

程序运行结果告诉我们,GET 提交方式不安全,连密码框都会显示在网址中。

9.2.3 表单的 POST 提交方式

为了获取通过 POST 方式提交的数据,需要使用 $_POST 变量。$_POST 变量是数组,数组的键是表单元素的 name 属性。例如,表单中有一个 name 属性为 text1 的文本框,获取该文本框的内容可以使用 $_POST['text1']。此外还要设置表单标签的 method 属性为 post。使用 POST 方式提交信息更安全。

表单的 POST 提交方式举例见前面的相关内容。

9.3 表单的高级操作

表单除了可以实现基本的数据处理之外,还可以实现诸如遍历表单元素、动态生成表单元素等操作。

9.3.1 表单元素的遍历

实际上表单元素可以看成表单数组,可以使用 foreach 循环来访问表单的每一个元素。

【示例 13】 表单元素的遍历。

eg13.php 代码如下。

```
<form name="form1" method="post" action="">
  姓名<input type="text" name="name" /><br />
  性别<input type="radio" name="sex" value="男" />男   
      <input type="radio" name="sex" value="女" />女<br />
  爱好<input type="checkbox" name="hobby[]" value="足球" />足球   
      <input type="checkbox" name="hobby[]" value="篮球" />篮球   
      <input type="checkbox" name="hobby[]" value="排球" />排球<br />
  <input type="submit" name="button" value="提交" />
</form>
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  if (isset($_POST['button'])){
    foreach($_POST as $name=>$value){
      echo "表单名:".$name.",表单值:";
      if (is_array($value)){
        echo implode(',',$value)."\n";
      }else{
        echo $value."\n";
      }
    }
  }
?>
```

程序运行结果如图 9-11 所示。左边是选择了而未提交,右边是提交后的结果。



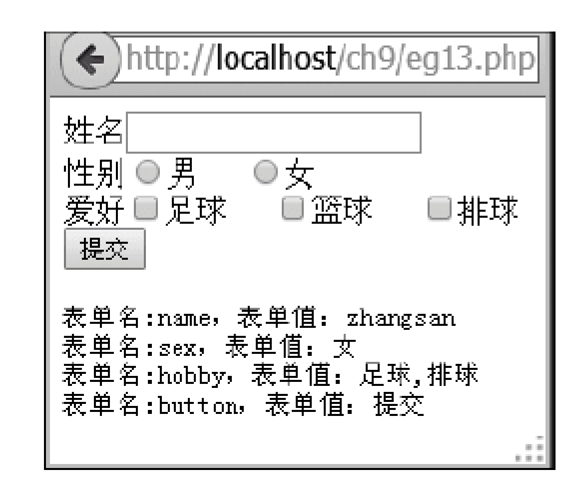


图 9-11　遍历表单元素

说明：表单元素实际上是名为 $_POST 的数组，而每一个表单元素则可以表示为 $_POST['name']，其中 name 是表单元素的名称属性。

而本例中作为爱好的多选框，其名称属性是 hobby[]，也是一个数组，因此在处理爱好表单元素时，通过 implode()函数来获取数组中每一个选中选项的值。

9.3.2　表单元素的动态生成

页面中有时候有一些数据不是一开始就存在，而是在后面动态产生的，有时候是从数据库中读取的。这些数据可以通过表单元素这样的载体来呈现。

表单元素主要有 3 个重要的属性，即 name、value 和默认值。因此在定义动态表单元素时只需要定义好这 3 个参数，即可动态产生一个表单元素。

1. 动态生成单选框

可以考虑把单选框的一些选项保存在数组中，再利用数组来动态创建单选框。

【示例 14】　表单元素单选框的动态创建。

eg14.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $tel =array(
    '010' =>'Beijing',
    '020' =>'Guangzhou',
    '021' =>'Shanghai',
    '023' =>'Chongqing',
    '027' =>'Wuhan',
  );
  $default ='023';                                          //默认选项
  echo "<form name='form1' method='post' action='' >";      //产生表单标签
  foreach($tel as $key=>$value){
    if ($key==$default)                                     //默认选中 key 为 023 的选项
      $checked ='checked';                                  //默认选中
    else
      $checked ='';                                         //不默认选中
    echo "<input type='radio' name='telofcity' value=\"$key\" $checked />$value\n";
  }
  echo "<input type='submit' name='button' value='提交' />";   //产生“提交”按钮
  echo "</form>";
```

```
  if (isset($_POST['button'])){
    echo "The tel of the city is:".$_POST['telofcity'];
  }
?>
```

程序运行结果如图 9-12 所示。左边是未选而并未提交，右边是选择 Shanghai 并单击“提交”按钮后的程序运行结果。





图 9-12　动态生成单选框

说明：foreach($tel as $key=>$value)用于遍历数组的每个元素，获取数组的 key 和 value 用作单选框的 value(选中后的返回值)和显示标题信息。

“echo "<input type='radio' name='telofcity' value=\"$key\" $checked />$value\n";”语句用于产生单选框。由于这些选项是一组单选框，故设置 name='telofcity'，保证多个选项的 name 属性是相同的；value=\"$key\"用于设置选中该选项后返回的值(即电话号码值)；$check 变量的值是空字符或'checked'，后者表示默认选中，仅有一个选项会默认选中；标签之外的 $value 用于设置单选框显示的标题信息；\n 用于换行显示不同选项。

2. 动态生成多选框

因为多选框可以多选，结果返回值是数组，故可以设置其 name 属性为数组，另外还可以使用 implode()函数来获取仍然将多选框各选项以及标题信息保存到数组中。

【示例 15】 表单元素多选框的动态创建。

eg15.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $hobby=array(
    'football'=>'足球',
    'basketball'=>'篮球',
    'run'=>'跑步',
    'swim'=>'游泳',
  );
  echo "<form name='form1' method='post' action=''>";          //产生表单标签
  foreach($hobby as $key=>$value){
     echo "<input type='checkbox' name='hobby[]' value=\"$key\" />$value ";
  }
  echo "\n<input type='submit' name='button' value='提交' />";  //产生“提交”按钮
  echo "</form>";
  if (isset($_POST['button'])){
```

```
    echo "Your hobbys are:".implode(',',$_POST['hobby']).'.';
  }
?>
```

程序运行结果如图 9-13 所示。左边选择了爱好而没有提交,右边是提交后的结果。





图 9-13　动态产生多选框

说明：$tel 数组保存着多个爱好的返回值和标题信息,其中数组的 key 作为选中该项的返回值,而 value 则作为该选项的标题信息。

foreach($hobby as $key=>$value)用于遍历数组元素,每个数组元素可以产生一个多选框;$key 和 $value 分别来自 $tel 数组的键和值。

"echo "<input type='checkbox' name='hobby[]' value=\"$key\" /> $value ";"语句用于产生多选框。因为多选框有多个返回值,故将这一组多选框的 name 设置为相同且是数组,即 name='hobby[]';value=\"$key\"表示选中该项返回值是数组的 key;多选框标签之外的 $value 用于设置多选框每个选项的标题文字;" "用于选项之间的空格。

3. 动态生成下拉列表框

下拉列表框既可以单选也可以多选,其实单选也可以视为特定的多选。仍然可以将下拉列表的选项、返回值等信息保存在数组中。再通过遍历数组元素生成多个下拉列表选项。

【示例 16】　表单元素下拉列表框的动态创建。

eg16.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $options =array(
    'sport' =>'体育',
    'art' =>'文艺',
    'society' =>'社会',
    'finance' =>'财经',
  );
  echo "<form name='form1' method='post' action=''>";          //产生表单标签
  echo "<select name='select1[]' multiple>";                   //产生下拉列表框
  foreach($options as $key=>$value){
    echo "<option value=\"$key\" >$value</option>";            //产生下拉列表选项
  }
  echo "</select>";
  echo "\n<input type='submit' name='button' value='提交' />";  //产生"提交"按钮
  echo "</form>";
  if (isset($_POST['button'])){
```

```
    echo "Your select(s) are(is):".@implode(',',$_POST['select1']).'.';
  }
?>
```

程序运行结果如图 9-14 所示。左边选择了选项但未提交，右边是提交后的运行结果。





图 9-14　动态生成下拉列表框

说明：$options 数组用于保存下拉列表框中的选项标题和返回值。

"foreach($options as $key=>$value)"语句用于遍历数组 $options 中的每个元素，数组的每个元素都是一个下拉列表选项。

"echo "<select name='select1[]' multiple>";"语句表示创建下拉列表框，考虑到允许多选，将 name 属性设置为数组，即"name='select1[]'"，而 multiple 则表示允许多选。

"echo "<option value=\"$key\" >$value</option>""语句用于产生下拉列表项，value=\"$key\"表示将数组 $options 的 key 作为下拉列表项的返回值，$value 则表示将数组的 value 作为下拉列表项的标题信息。

9.4　综合案例——用户注册

用户注册在 Web 系统中是常见的功能。本节将以用户注册来讲解表单的设计、信息的提交和输出。

本注册程序需要用户从页面提交用户名、密码、性别、爱好、班级、备注等信息，要求信息提交后在页面中显示这些信息。

步骤 1　启动 DW，新建 eg17.php 网页文件，并进行界面设计。添加表单，设置表单属性：<form name="form1" method="post" action="">。在表单中再添加表格，表格为 7 行 2 列，将第 4 行、第 5 行、第 7 行的两列合并为一列。在表格中添加表单元素：文本框、密码框、单选框、复选框、下拉列表框和多行文本框、"提交"按钮，如图 9-15 所示。

eg17.php 代码如下。

```
<meta http-equiv="content-type" content="text/html; charset=UTF-8" />
<form name="form1" method="post" action="">
  <table width="480" border="1" align="center">
    <tr>
      <td>用户名
        <input type="text" name="user" >
      </td>
```

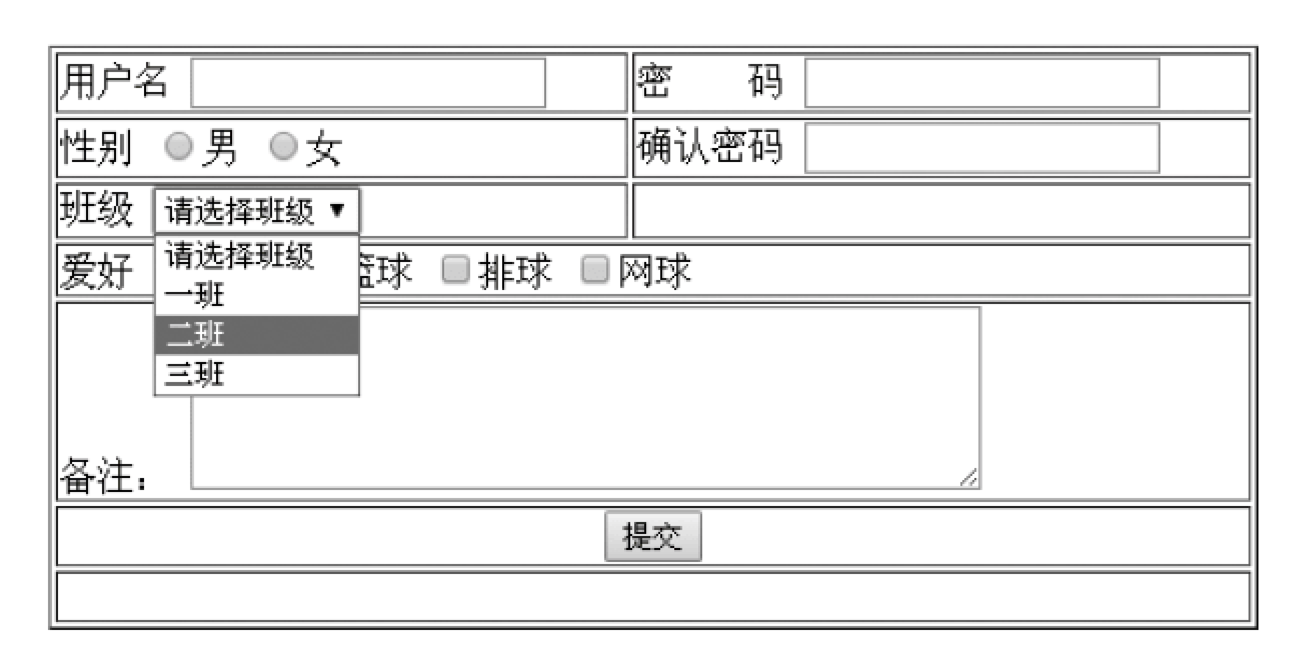


图 9-15　注册页面设计

```
  <td>密     码
    <input type="password" name="pa1" id="p1">
  </td>
</tr>
<tr>
  <td>性别
    <input type="radio" name="sex" value="男">男
    <input type="radio" name="sex" value="女">女
  </td>
  <td>确认密码
    <input type="password" name="pa2" id="p2">
  </td>
</tr>
<tr>
  <td>班级
    <select name="select" id="select" >
      <option value="">请选择班级</option>
      <option value="一班">一班</option>
      <option value="二班">二班</option>
      <option value="三班">三班</option>
    </select>
   </td>
  <td>  </td>
</tr>
<tr>
  <td colspan="2">
    爱好
    <input type="checkbox" name="hobby[]" value="足球">足球
    <input type="checkbox" name="hobby[]" value="篮球">篮球
    <input type="checkbox" name="hobby[]" value="排球">排球
    <input type="checkbox" name="hobby[]" value="网球">网球
 </td>
</tr>
<tr>
  <td colspan="2">备注：
    <textarea name="notice" cols="45" rows="5" ></textarea>
  </td>
</tr>
```

```
        <tr>
          <td colspan="2" align="center">
            <input type="submit" name="button" value="提交" >
          </td>
        </tr>
        <tr>
          <td colspan="2" align="left"> 
          </td>
        </tr>
      </table>
    </form>
```

步骤 2 为“提交”按钮编写代码，单击“提交”按钮后，在表格最后一行显示注册信息。代码如下(见粗体字)：

```
<meta http-equiv="content-type" content="text/html; charset=UTF-8" />
<form name="form1" method="post" action="">
  <table width="480" border="1" align="center">
    <tr>
      <td>用户名
        <input type="text" name="user" >
      </td>
      <td>密     码
        <input type="password" name="pa1" id="p1">
      </td>
    </tr>
    <tr>
      <td>性别
        <input type="radio" name="sex" value="男">男
        <input type="radio" name="sex" value="女">女
      </td>
      <td>确认密码
        <input type="password" name="pa2" id="p2">
      </td>
    </tr>
    <tr>
      <td>班级
        <select name="select" id="select" >
          <option value="">请选择班级</option>
          <option value="一班">一班</option>
          <option value="二班">二班</option>
          <option value="三班">三班</option>
        </select>
       </td>
      <td> </td>
    </tr>
    <tr>
      <td colspan="2">
        爱好
        <input type="checkbox" name="hobby[]" value="足球">足球
        <input type="checkbox" name="hobby[]" value="篮球">篮球
```

```
        <input type="checkbox" name="hobby[]" value="排球">排球
        <input type="checkbox" name="hobby[]" value="网球">网球
    </td>
  </tr>
  <tr>
    <td colspan="2">备注：
      <textarea name="notice" cols="45" rows="5" ></textarea>
    </td>
  </tr>
  <tr>
    <td colspan="2" align="center">
      <input type="submit" name="button" value="提交" >
    </td>
  </tr>
  <tr>
    <td colspan="2" align="left">
     <?php
       if (isset($_POST['button'])){
          echo "用户名:".$_POST['user']."<br />";
          echo "密   码:".$_POST['pa1']."<br />";
          echo "性   别:".$_POST['sex']."<br />";
          echo "班   级:".$_POST['select']."<br />";
          echo "爱   好:".@implode(',',$_POST['hobby'])."<br />";
          echo "备   注:".$_POST['notice'];
       }
     ?> 
    </td>
  </tr>
 </table>
</form>
```

程序运行结果如图 9-16 所示。



图 9-16　“用户注册”程序运行结果

步骤 3　改进程序。从运行效果看出，当用户单击“提交”按钮后，注册信息不再显示在表单中，且还需要验证两次输入的密码是否相同。

关于表单元素在单击“提交”按钮之后仍然保持选中或者填写状态的解决方法。

(1) 文本框、密码框、多行文本框问题的解决

在表单提交之后，表单中的用户名、密码、备注信息的值应该为 $_POST['user']、$_POST['pass1']、$_POST['notice']。而在提交之前表单中的用户名、密码、备注信息的值则为空。综合二者，可以为这三个表单元素设置初始 value 属性：

```
<input type="text" name="user" value="<?php echo @$_POST['user'];?>">
<input type="password" name="pass1" value="<?php echo @$_POST['pass1'];?>">
<textarea name="notice"cols="45"rows="5"><?php echo @$_POST['notice'];?>
</textarea>
```

说明：由于上述代码设置了 value 属性，因此在表单提交后，在表单元素中仍然能够显示表单元素的内容。而@符号的作用在于：在表单提交前表单元素的值是不存在的，此时使用 echo 输出表单元素值会产生 notice 系统信息，@则可以过滤这种 notice 系统信息。

(2) 单选框在提交之后继续保持原来的选中状态的解决方法

在表单中，性别使用单选框来显示，可以在“男”和“女”中选择其中之一。我们希望在表单提交之后，单选框继续保持原来的选中状态。实际上在表单提交后，$_POST['sex']的值必为“男”或“女”，若为“男”，则选中第一个单选框；若为“女”，则选中第二个单选框。若 $_POST['sex']的值既不是“男”也不是“女”，则一个也不选中。初始状态就是一个也不选中。根据分析，改进的代码如下。

```
<input type="radio" name="sex" value="男" <?php if (@$_POST['sex']=='男') echo
'checked';?>>男
<input type="radio" name="sex" value="女" <?php if (@$_POST['sex']=='女') echo
'checked';?>>女
```

说明：在上述代码中，两个单选框的 name 属性都为 sex，这两个单项框是一组，因此两个 if 语句不可能同时成立，程序只会选中其中一种性别。此外，@符号可以避免在提交之前因 $_POST['sex']不存在而显示 notice 系统信息。

(3) 下拉列表框在选择班级并提交表单之后仍然保持班级选中状态的解决方法

可以判断每一个选项是否等于 $_POST['select']，若某一个选项的 value 值刚好等于 $_POST['select']，则选中该项。根据分析，改进代码为：

```
<select name="select" id="select">
    <option value="">请选择班级</option>
    <option value="一班" <?php if (@$_POST['select']=='一班') echo 'selected';?>>一班
      </option>
    <option value="二班" <?php if (@$_POST['select']=='二班') echo 'selected';?>>二班
      </option>
    <option value="三班" <?php if (@$_POST['select']=='三班') echo 'selected';?>>三班
      </option>
</select>
```

说明：在表单提交之前，默认会选中第一项(即<option value="">请选择班级</option>)；在表单提交后，若 $_POST['select']等于'一班'，则会选中'一班'这一项，其他班级相同。三个班级不可能同时选中，三个 if 语句也不会同时成立。

(4) 复选框在选择爱好之后继续保持选中状态的解决方法

复选框采用数组来解决。四个复选框的 name 属性都相同：name="hobby[]"。我们假设选中了“足球”和“篮球”。则表单提交后 $_POST['hobby']为数组，即 array("足球","篮球")；此时判断得知足球被选中，于是选中该复选框，即“if (@in_array('足球',@$_POST['hobby'])) echo "checked";”；同理也会选中“篮球”选项。根据分析改进代码为：

```
<input type="checkbox" name="hobby[]" value="足球"
  <?php if (@in_array('足球',@$_POST['hobby'])) echo "checked";?>>足球
<input type="checkbox" name="hobby[]" value="篮球"
  <?php if (@in_array('篮球',@$_POST['hobby'])) echo "checked";?>>篮球
<input type="checkbox" name="hobby[]" value="排球"
  <?php if (@in_array('排球',@$_POST['hobby'])) echo "checked";?>>排球
<input type="checkbox" name="hobby[]" value="网球"
  <?php if (@in_array('网球',@$_POST['hobby'])) echo "checked";?>>网球
```

说明：复选框可以同时选中多项，因此表单提交后可以有多项被选中。@作用同前面。

(5) 两次输入相同密码问题的解决方法

使用 JavaScript 方法，为“提交”按钮添加属性如下。

```
<input type="submit" name="button" value="提交" onclick="return chick();" >
```

并编写 JavaScript 代码：

```
<script language="javascript">
  function chick(){
   if(document.getElementById("p1").value==document.getElementById("p2").value)
     return true;
   else{
     alert('两次密码不一样，请重新输入！');
     document.getElementById('p1').focus();
     return false;
   }
  }
</script>
```

在上述代码中，若两个密码框中输入的密码不相同，则显示提示对话框“两次密码不一样，请重新输入！”，并将焦点放到第一个密码框(id 为 p1 的那个密码框)，并返回 false。“提交”按钮在单击之后返回 false，是不会触发表单的提交行为的。若两次输入的密码相同，则返回 true，此时触发表单的提交行为。

(6) 为表单元素添加 required 属性，解决某表单元素在提交前必须填写的问题

例如，文本框对应的用户名：

```
<input type="text" name="user" value="<?php echo @$_POST['user'];?>" required>
```

其他表单元素省略。

改进后的完整代码 eg7-2.php 如下。

```
<meta http-equiv="content-type" content="text/html; charset=UTF-8" />
<script language="javascript">
```

```
function chick(){
 if(document.getElementById("p1").value==document.getElementById("p2").value)
   return true;
 else{
   alert('两次密码不一样,请重新输入!');
   document.getElementById('p1').focus();
   return false;
 }
}
</script>
<form name="form1" method="post" action="">
  <table width="480" border="1" align="center">
    <tr>
      <td>用户名
        <input type="text"name="user"required value="<?php echo @$_POST
        ['user'];?>">
      </td>
      <td>密     码
        <input type="password" name="pa1" id="p1"
          value="<?php echo @$_POST['pa1'];?>" required>
      </td>
    </tr>
    <tr>
      <td>性别
        <input type="radio" name="sex" value="男" required
          <?php if (@$_POST['sex']=='男') echo 'checked';?>>男
        <input type="radio" name="sex" value="女" required
          <?php if (@$_POST['sex']=='女') echo 'checked';?>>女
      </td>
      <td>确认密码
        <input type="password" name="pa2" id="p2"
          value="<?php echo @$_POST['pa2'];?>" required>
      </td>
    </tr>
    <tr>
      <td>班级
        <select name="select" id="select" required>
          <option value="">请选择班级</option>
          <option value="一班" <?php if (@$_POST['select']=='一班')
            echo 'selected';?>>一班</option>
          <option value="二班" <?php if (@$_POST['select']=='二班')
            echo 'selected';?>>二班</option>
          <option value="三班" <?php if (@$_POST['select']=='三班')
            echo 'selected';?>>三班</option>
        </select>
      </td>
      <td> </td>
```

```
    </tr>
    <tr>
      <td colspan="2">
        爱好
        <input type="checkbox" name="hobby[]" value="足球"
          <?php if (@in_array('足球',@$_POST['hobby'])) echo "checked";?>>足球
        <input type="checkbox" name="hobby[]" value="篮球"
          <?php if (@in_array('篮球',@$_POST['hobby'])) echo "checked";?>>篮球
        <input type="checkbox" name="hobby[]" value="排球"
          <?php if (@in_array('排球',@$_POST['hobby'])) echo "checked";?>>排球
        <input type="checkbox" name="hobby[]" value="网球"
          <?php if (@in_array('网球',@$_POST['hobby'])) echo "checked";?>>网球
     </td>
    </tr>
    <tr>
      <td colspan="2">备注：
        <textarea name="notice" cols="45" rows="5"
        required><?php echo @$_POST['notice'];?></textarea>
      </td>
    </tr>
    <tr>
      <td colspan="2" align="center">
        <input type="submit" name="button" value="提交" onclick="return chick();" >
      </td>
    </tr>
    <tr>
      <td colspan="2" align="left">
       <?php
         if (isset($_POST['button'])){
            echo "用户名:".$_POST['user']."<br />";
            echo "密   码:".@$_POST['pal']."<br />";
            echo "性   别:".@$_POST['sex']."<br />";
            echo "班   级:".@$_POST['select']."<br />";
            echo "爱   好:".@implode(',',$_POST['hobby'])."<br />";
            echo "备   注:".@$_POST['notice'];
         }
       ?> 
      </td>
    </tr>
  </table>
</form>
```

程序运行结果如图 9-17 所示。



图 9-17 改进后程序的运行结果

9.5 习 题

一、填空题

1. 表单的 method 属性值可以是 get 或________。

2. 表单提交时使用________提交方式可将数据显示在 URL 上。

3. 多个单选框分为一组，可以将这些单选框的________属性设置为相同。

4. 将多选框的________属性设置为同名的数组，可以采用数组的方式批量处理。

5. 用 POST 方法提交表单时，在 PHP 脚本中必须使用________变量获取表单中的数据。

6. 密码框的定义使用 type=________来设置。

二、选择题

1. 添加多行文本框使用________语句。

A. <textarea name="..."></textarea>

B. <input type="textarea" ...></textarea>

C. <input type="text">...</textarea>

D. <text name="..."</text>

2. 下列关于隐藏域说法正确的是________。

A. 隐藏域在页面初始化时隐藏，在某个事件引发显示

B. 隐藏域用来获取从数据或脚本中读取的数据，并在获取数据后显示

C. 隐藏域是不会显示出来的

D. 隐藏域保存的数据是可有可无的

3. 下列关于下拉列表框说法错误的是________。

A. 下拉列表框的 type 值为 dropdownbox

B. 下拉列表框可以选择一个或多个选项

C. 下拉列表框可以有一个或多个选项

D. 下拉列表框可以设置默认选项

4. 读取 POST 方法传递的表单元素的方法是________。

A. $_post["名称"]　　　　B. $_POST["名称"]

C. post["名称"]　　　　D. $POST["名称"]

5. 下列说法错误的是________。

A. "提交"按钮用于提交表单数据

B. "重置"按钮用于将表单元素的值清零

C. 一般按钮用于执行 PHP 脚本

D. 一般按钮用于执行 JavaScript 代码

三、程序设计题

下面是一个班上的考试信息，编写程序在文本框中输入 ID，单击"查询"按钮之后，能够在表单中显示该考生的个人信息，如图 9-18 所示。

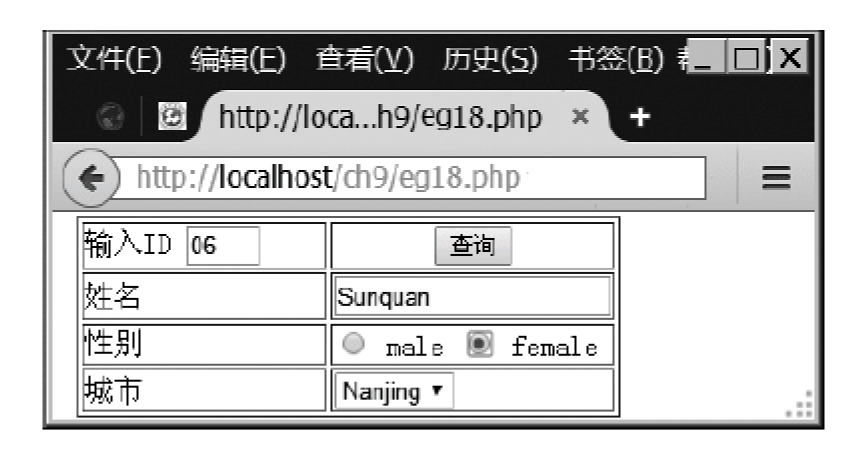


图 9-18　信息查询程序的运行结果

```
$arr =array( array('id'=>'01','xingming'=>'Liubei','sex'=>'male' ,'city'=>'Chengdu'),
             array('id'=>'02','xingming'=>'Guanyu','sex'=>'male' ,'city'=>'Chengdu'),
             array('id'=>'03','xingming'=>'Yuanshu','sex'=>'male','city'=>'Shouchun'),
             array('id'=>'04','xingming'=>'Caocao','sex'=>'female' ,'city'=>'Xuchang'),
             array('id'=>'05','xingming'=>'Liuxie','sex'=>'male' ,'city'=>'Changan'),
             array('id'=>'06','xingming'=>'Sunquan','sex'=>'female','city'=>'Nanjing')
);
```

第 10 章　session 和 cookie

知识点：

- session 的基本知识
- cookie 的基本知识
- session 的配置
- cookie 的工作原理
- session 的基本操作
- cookie 的基本操作

本章导读：

有过购物经历的用户应该知道，在我们购买商品时，用户挑选商品并放置到购物车中，这些功能实际上是通过 session 会话来实现的。使用邮箱的用户还能发现，下次登录邮箱时发现用户名不需要输入，甚至有时候连密码也不用输入，这实际上是通过 cookie 来实现的。本章将讲述 session 和 cookie 的工作原理和操作方法等知识点。

10.1　session 的基本知识

session 用于浏览器和服务器之间的会话，并将会话信息保存在服务器上。每一次浏览器和服务器之间的会话是有生命周期的，且服务器能够记录不同会话的 Session ID。不同的浏览器甚至相同的浏览器在不同时间段与服务器的会话都视为不同的会话，在服务器上使用不同的 Session ID。使用 session 可以在不同网页之间，或在网页与服务器之间传递信息。

10.1.1　session 简介

session 的中文译名叫作“会话”，其本来的含义是指有始有终的一系列动作/消息，比如打电话时从拿起电话拨号到挂断电话这中间的一系列过程可以称为一个 session。目前社会上对 session 的理解非常混乱：有时候我们可以看到这样的话：“在一个浏览器会话期间……”这里的会话是指从一个浏览器窗口打开再到关闭这个期间；也可以看到“用户（客户端）在一次会话期间”这样一句话，它可能指用户的一系列动作。一般情况下是同某个具体目的相关的一系列动作，比如从登录到选购商品到结账这样一个网上购物的过程；有时候也可能仅仅是指一次连接，其中的差别只能靠上下文来推断了。

然而当 session 一词与网络协议相关联时，它又往往隐含了“面向连接”和/或“保持状态”这样两个含义，“面向连接”是指在通信双方在通信之前要先建立一个通信的渠道，比如

打电话，直到对方接了电话通信才能开始。“保持状态”则是指通信的一方能够把一系列的消息关联起来，使消息之间可以互相依赖，比如一个服务员能够认出再次光临的老顾客并且记得上次这个顾客还欠店里一块钱。这一类的例子有“一个 TCP session”或者“一个 POP3 session”。

鉴于这种混乱已不可改变，要为 session 下一个定义就很难有统一的标准。而在阅读 session 相关资料时，我们也只有靠上下文来推断理解了。不过可以这样理解：假如我们打电话，从拨通的那一刻起到挂断电话期间，因为电话一直保持着接通的状态，所以把这种接通的状态叫作 session。它是访客与整个网站交互过程中一直存在的公有变量，在客户端不支持 cookie 时，为了保证数据正确、安全，就采用 session 变量。访问网站的来客会被分配一个唯一的标识符，即所谓的会话 ID。它要么存放在客户端的 cookie，要么经由 URL 传递。

session 的发明填补了 HTTP 协议的局限：HTTP 协议被认为是无状态协议，无法得知用户的浏览状态，当它在服务端完成响应之后，服务器就失去了与该浏览器的联系。这与 HTTP 协议本来的目的是相符的，客户端只需要简单地向服务器请求下载某些文件，无论是客户端还是服务器，都没有必要记录彼此过去的行为，每一次请求之间都是独立的，好比一个顾客和一个自动售货机或者一个普通的(非会员制)大卖场之间的关系一样。

因此通过 session(cookie 是另外一种解决办法)记录用户的有关信息，以供用户再次以此身份对 Web 服务器提起请求时作确认。会话的发明使一个用户在多个页面间切换时能够保存他的信息。网站编程人员都有这样的体会，每一页中的变量是不能在下一页中使用的(虽然 form、URL 也可以实现，但这都是非常不理想的办法)，而 session 中注册的变量就可以作为全局变量使用了。

那么 session 到底有什么用处呢？网上购物时大家都用过购物车，你可以随时把自己选购的商品加入到购物车中，最后再去收银台结账。在整个过程中购物车一直扮演着临时存储被选商品的角色，用它追踪用户在网站上的活动情况，这就是 session 的作用，它可以用于用户身份认证、程序状态记录、页面之间参数传递等。

session 的实现中采用了 cookie 技术，session 会在客户端保存一个包含 session_id (session 编号)的 cookie；在服务器端保存其他 session 变量，比如 session_name 等。当用户请求服务器时也把 session_id 一起发送到服务器，通过 session_id 提取所保存在服务器端的变量，就能识别用户是谁。同时也不难理解为什么 session 有时会失效了。

10.1.2　session 配置

在 PHP 中 session 会话的配置通过配置文件 php.ini 来实现。该文件的位置一般在文件夹 D:\phpStudy\php\php-5.4.45 中，读者的配置文件的位置与你的安装路径和版本有关。下面列出了 PHP 中与 session 有关的主要配置项(“等于号”之后的值是默认值)。

1. session.save_path

该项用于设置 session 文件的保存位置。可以使用“*N*;[MODE;]/path”这样的模式定义该路径，*N* 是一个整数，表示使用 *N* 层深度的子目录，而不是将所有数据文件都保存在一个目录下。“[MODE;]”可选，必须使用八进制数，默认为 600(=384)，表示每个目录下最多保存的会话文件数量。“[MODE;]”并不会改写进程的 umask。PHP 不会自动创建这些

文件夹结构,可使用 ext/session 目录下的 mod_files. sh 脚本创建。如果该文件夹可以被不安全的用户访问(比如默认的/tmp),那么将会带来安全漏洞。当 $N>0$ 时自动垃圾回收将会失效,具体参见下面有关垃圾收集的部分。

2. session. save_handler ="files"

默认以文件方式存取 session 数据,如果想要使用自定义的处理器来存取 session 数据,比如数据库,可用“user”。

3. session. use_cookies = 1

确定是否使用 cookies 在客户端保存会话 sessionid,默认为采用 cookies。

4. session. use_only_cookies = 0

确定是否仅仅使用 cookie 在客户端保存会话 sessionid。这个选项可以使管理员禁止用户通过 URL 来传递 id,默认为 0。客户端如果禁用 cookie,将使 session 无法工作。

5. session. name = "PHPSESSID"

当作 cookie name 来使用的 session 标识名。

6. session. auto_start = 0

确定是否自动启动 session,默认不启动。我们知道在使用 session 功能时,基本上在每个 php 脚本头部都会通过 session_start()函数来启动 session,如果启动了这个选项,则在每个脚本头部都会自动启动 session,不需要每个脚本头部都以 session_start()函数启动 session。推荐关闭这个选项,采用默认值。

7. session. cookie_lifetime = 0

传递 sessionid 的 cookie 有效期(秒),0 表示仅在浏览器打开期间有效。

8. session. gc_divisor = 100

定义在每次初始化会话时启动垃圾回收程序的概率。计算公式如下。session. gc_probability/session. gc_divisor,比如 1/100,表示有 1%的概率启动垃圾回收程序,对会话页面访问越频繁,概率就应当越小。建议值为 1/1000~1/5000。

9. session. gc_maxlifetime = 1440

设定保存的 session 文件生存期,超过此参数设定秒数后,保存的数据将被视为“垃圾”并由垃圾回收程序清理。判断标准是最后访问数据的时间(对于 FAT 文件系统是最后刷新数据的时间)。如果多个脚本共享同一个 session. save_path 目录但 session. gc_maxlifetime 不同,将以所有 session. gc_maxlifetime 指令中的最小值为准。

本节列出了关于 session 的一些配置。更多更详细的配置,读者可以参阅相关手册。

10.1.3 session 函数

PHP 中提供了一系列与 session 有关的函数,通过这些函数可以启动 session,也可以删除 session,还可以设置 10.1.2 小节讲述的配置项。表 10-1 列出了常用的与 session 有关的函数。

表 10-1 常用的与 session 有关的函数

函数名称	说明
session_start()	启动 session

续表

函 数 名 称	说　　明
session_destroy()	销毁已注册的所有 session
session_name()	设置或获取一个 session name
session_module_name()	设置或获取一个 session 的存储形式
session_save_path()	设置或获取 session 在服务器中的保存位置
session_id()	定义或获取当前的 Session ID
session_register()	将一些字符串或数组注册到 session 中
session_unregister()	取消一个已注册的 session
session_is_registered()	检测一个 session 是否已注册
session_decode()	将序列号的 session 数据还原
session_encode()	序列化当前 session 中的所有数据
session_commit()	终止由 session_start()函数开启的 session 可写入状态，同 session_write_close()
session_cache_expire()	设置或取消 session 过期的时间
session_cache_limiter()	设置或获取 session 缓存的类型
session_write_close()	终止由 session_start()函数开启的 session 可写入状态，同 session_commit()

10.1.4　session 变量

session 变量是全局变量，可以将 session 变量理解为一个数组，该变量映射了 session 生命周期的 session 数据，并保存在内存中。在 session 初始化时，从 session 文件中读出数据并保存到该 session 变量中。session 生命周期结束时，将 session 数据写回 session 文件。session 变量的书写方式为 $_SESSION['name']。

【示例 1】　session 变量的使用。

eg1.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  session_start();              //启动 session
  if (isset($_SESSION['user']))  //判断$_SESSION['user']是否存在，第一次是不存在的
    $_SESSION['counter']++;     //若不是第一次访问，则次数自动加 1
  else{
  //若第一次访问，则创建$_SESSION['counter']和$_SESSION['user']两个变量，并赋初值
    $_SESSION['counter'] =1;
    $_SESSION['user'] ='liubei';
  }
  echo "欢迎{$_SESSION['user']}，你是第{$_SESSION['counter']}次来访！"
?>
```

程序运行结果如图 10-1 所示。左边是第一次运行，右边是多次刷新之后的结果。

说明：一次 session 会话，从启动浏览器运行服务器中的网页开始，关闭浏览器则会话结束。仅关闭本页面而不关闭浏览器仍然视为同一次的会话。



图 10-1　session 变量应用举例

“isset($_SESSION['user']用于判断是否存在 $_SESSION['user'],若存在则表明不是第一次登录,此时仅需要把访问次数 $_SESSION['counter']加 1 即可;若不存在则表明是第一次登录,此时需要创建两个 session 变量: $_SESSION['counter']和 $_SESSION['user'],并为之赋初始值。

10.2　session 的基本操作

session 的基本操作包括 session 的启动、销毁和数据的存储等。

10.2.1　session 的启动

在把用户信息存储到 session 之前,必须先启动 session。在 PHP 中可以使用 session_start()函数来启动 session。语法格式为:

```
bool session_start(void)
```

session_start()函数用于创建一个新会话或者继续当前的会话。

此外,还可以通过设置 php.ini 文件中的 session.auto_start 选项来启动 session。将 session.auto_start 设置为 1(默认值为 0,0 表示不启动 session),然后重启服务,即可以启动 session,并且每一个 PHP 页面都会自动启动 session。而使用 session_start()函数启动 session 则只能作用到调用该函数的 PHP 页面。

10.2.2　sessionID 的获取

在 PHP 脚本中第一次调用 session_start()时将产生一个 sessionID,这个 ID 将全部会话数据绑定到某个特定用户。虽然 PHP 能够创建和传播 sessionID,但并不是在任何情况下都希望使用 PHP 自动创建和传播 ID,有时候也需要手工获取和设置 sessionID,这时可以使用 session_id()函数。

session_id()函数用于显示 sessionID。sessionID 是一个 32 位的字符串,是用户使用 session 时的钥匙。该函数的语法格式为:

```
string session_id([string id])
```

该函数有一个可选参数。若不指定该参数,则表示获取当前的 sessionID;若指定参数,则表示使用指定的参数替换当前的 sessionID。

【示例 2】　sessionID 的获取或设置。

eg2.php 代码如下。

```
<?php
```

```
    session_start();
    echo session_id()."<br />";   //获取当前的 SessionID
    session_id('userid001');      //设置当前的 SessionID 为 userid001
    echo session_id();            //输出当前的 SessionID
?>
```

程序运行结果如图 10-2 所示。



图 10-2 SessionID 的获取或设置

10.2.3 session 的存取

存储和读取 session 变量的正确方法是使用 PHP 的 session 变量，它是 PHP 全局变量，用来存储 session 会话。使用 $_SESSION 关联数组来表示 session 变量。$_SESSION 关联数组的键名与变量命名规则相同。

session 不仅可以存取普通变量，还可以存取数组和对象。普通变量的存取见示例 1。

【示例 3】 使用 session 存取数组。

eg3.php 代码如下。

```
<?php
    session_start();
    $user =array(
                'no'=>'0001',
                'name'=>'liubei',
                'sex'=>'male',
                'age'=>23,
            );
    $_SESSION['user'] =$user;
    echo "<pre>";
    foreach($_SESSION['user'] as $key=>$value)
        echo $key.'---'.$value."\n";
?>
```

程序运行结果如图 10-3 所示。



图 10-3 使用 session 存取数组

【示例 4】 使用 session 存取对象。

eg4. php 代码如下。

```
<?php
  class teacher{
    private $no,$name,$sex,$age;
    function __construct($no,$name,$sex,$age){
      $this->no =$no;
      $this->name =$name;
      $this->sex =$sex;
      $this->age =$age;
    }
    function showinfo(){
      echo "<pre>";
      echo $this->no."\n";
      echo $this->name."\n";
      echo $this->sex."\n";
      echo $this->age;
    }
  }
  $t1 =new teacher('0001','liubei','male','23');
  $_SESSION['t1'] =$t1;             //$_SESSION['t1']是一个对象
  $_SESSION['t1']->showinfo();      //调用 teacher 类的对象的方法
?>
```

程序运行结果如图 10-4 所示。

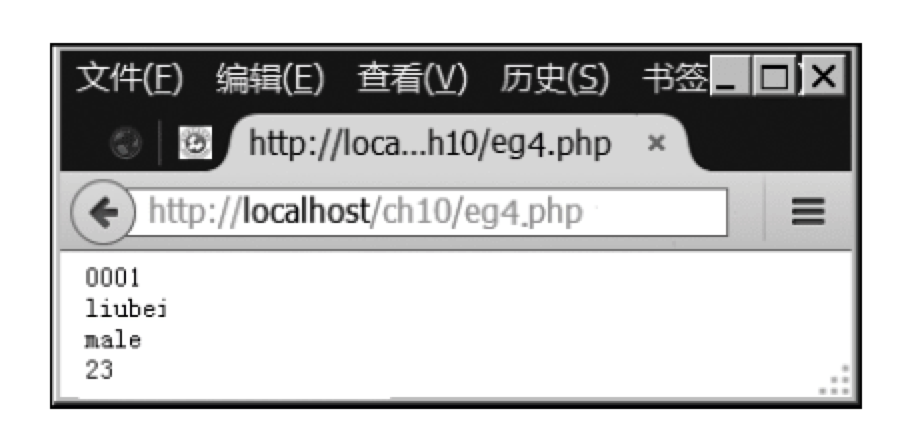


图 10-4 使用 session 存取对象

10.2.4 session 的销毁

如果关闭浏览器，则当前会话自动中断，打开新浏览器时会自动创建一个新的会话。但是在实际应用中需要使用程序中断会话或者删除会话变量，例如单击某网站的“安全退出”按钮，系统需要清除 session 信息，以保证用户信息安全。当会话不再使用时，需要人为地销毁它。虽然 PHP 能够自动销毁会话，但是效率比较低，灵活性不够。下面将讲述多种人为地销毁会话的函数。

1. unset()函数

PHP 中提供的 unset()函数用于删除指定的变量，当然还包括已经建立的 $_SESSION 变量，但不会删除 session 文件以及不释放 SessionID。unset()函数销毁会话变量的形式为：

```
unset($_SESSION[key])
```

key 是 $_SESSION 变量(数组)的键。

【示例 5】 使用 unset()函数销毁会话变量。

eg5.php 代码如下。

```
<?php
  session_start();
  $_SESSION['name']='liubei';
  $_SESSION['age']=23;
  unset($_SESSION['name']);
  echo $_SESSION['age'];          //显示 23
  echo $_SESSION['name'];
  //因为$_SESSION['name']会话变量被释放,所以显示下面的信息
  //Notice: Undefined index: name in D:\phpStudy\WWW\ch10\eg5.php on line 7
?>
```

2. session_unset()函数

session_unset()函数用于清除存储在当前会话中的全部 session 变量,语法形式为:

```
void session_unset();
```

【示例 6】 使用 session_unset()函数销毁会话变量。

eg6.php 代码如下。

```
<?php
  session_start();
  $_SESSION['name']='liubei';
  $_SESSION['age']=23;
  session_unset();               //清除全部 session 变量
  echo $_SESSION['age'];
  //Notice: Undefined index: age in D:\phpStudy\WWW\ch10\eg6.php on line 6
  echo $_SESSION['name'];
  //Notice: Undefined index: age in D:\phpStudy\WWW\ch10\eg6.php on line 8
  echo session_id();             //输出 session_id()
?>
```

3. session_destroy()函数

session_destroy()函数结束当前的会话,并清空会话中的所有资源。该函数不会释放(unset)与当前 session 相关的全局变量(globalvariables),也不会删除客户端的 session cookie。PHP 默认的 session 是基于 cookie 的,如果要删除 cookie,必须借助 setcookie()函数。

【示例 7】 使用 session_destroy()函数结束当前的会话。

eg7.php 代码如下。

```
<?php
  session_start();
  $_SESSION['name']='liubei';
  $_SESSION['age']=23;
  session_destroy();
  echo $_SESSION['age'];     //输出$_SESSION['age']
```

```
  echo $_SESSION['name'];     //输出$_SESSION['name']
  echo session_id();          //不会输出 session_id()
?>
```

10.3 session 举例

使用 session 变量实现网站登录。

【示例 8】 使用 session 变量实现网站登录。

创建两个页面 login.php 页面和 main.php 页面。其中 login.php 页面用于输入用户名和密码，当用户名和密码都正确时即可进入 main.php 页面，否则会显示用户名或者密码错误；若直接运行 main.php 页面，则强行跳转到登录页面 login.php。

思路：可以考虑在登录页面中的用户名和密码都输入正确时创建两个 session 变量，分别保存用户名和密码，网页跳转到 main.php 页面时这两个 session 变量不会销毁，因此可以通过检测这两个 session 变量来判断用户的合法性，若用户名和密码正确则给出欢迎信息，否则强行跳转到 login.php 页面。

步骤 1 创建登录页面 login.php。

login.php 代码如下。

```
<form name="form1" method="post" action="">
用户名<input type="text" name="user" /><br />
密   码<input type="password" name="password" /><br />
<input type="submit" name="button1" value="登录" />
</form>
<?php
  header("content-type:text/html;charset=utf-8");
  session_start();
  if (isset($_POST['button1'])){
    if (($_POST['user']=='liubei') && ($_POST['password']=='123456')){
      $_SESSION['user'] =$_POST['user'];
      $_SESSION['password'] =$_POST['password'];
      echo "<script>window.location.href='main.php';</script>";
    }else
      echo "<script>alert('用户名或密码错误!');</script>";
  }
?>
```

说明：程序登录页面 login.php 在运行时若输入用户名和密码分别是 liubei 和 123456 并单击“登录”按钮，则会跳转到 main.php 页面，并且把用户名和密码分别保存在两个 session 变量 $_SESSION['user']和 $_SESSION['password']中，这样便可以在 main.php 页面中进行用户名和密码的核对。

若输入的用户名和密码有一个不匹配，则会弹出对话框显示“用户名或密码错误!”，也不跳转到 main.php 页面。

步骤 2　创建 main. php 页面。该页面负责核对两个 session 变量 $ _SESSION['user'] 和 $ _SESSION['password']是否存在,若存在,还要核对其值是否是登录页面 login. php 中输入的用户名和密码,若是,则显示欢迎信息。若两个 session 变量 $ _SESSION['user'] 和 $ _SESSION['password']不存在或者即使存在但是它们的值不是 login. php 页面中输入的用户名和密码,则程序仍然强行跳到登录页面 login. php。

eg8/main. php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  session_start();
  if (isset($_SESSION['user']) && isset($_SESSION['password']))
    if (($_SESSION['user']=='liubei') && ($_SESSION['password']=='123456'))
      echo $_SESSION['user'].",you are welcome!";  //若用户名和密码都对,显示欢迎信息
    else     //若用户名或密码不匹配,则跳转到登录页面 login.php
      echo "<script>window.location.href='login.php';</script>";
  else       //若不存在 session 变量,则跳转到登录页面 login.php
     echo "<script>window.location.href='login.php';</script>";
?>
```

说明:若核对发现两个 session 变量并不存在,则用户肯定是未经合法登录,直接运行 main. php 页面,程序会强行跳转到登录页面 login. php,强行要求登录。

若核对发现两个 session 变量存在,但是保存用户名和密码的两个 session 变量值并不与登录页面中输入的用户名和密码相同,则仍然视为非法用户,程序强行跳转到 login. php 登录页面。

10.4　cookie 的基本知识

cookie 现在经常被大家提到,那么到底什么是 cookie? 它有什么作用呢? cookie 是一种能够让网站服务器把少量数据储存到客户端的硬盘或内存,或是从客户端的硬盘读取数据的一种技术。cookie 是当你浏览某网站时,由 Web 服务器置于你硬盘上的一个非常小的文本文件,它可以记录你的用户 ID、密码、浏览过的网页、停留的时间等信息。当你再次来到该网站时,网站通过读取 cookie,得知你的相关信息,就可以做出相应的动作,如在页面显示欢迎你的标语,或者让你不用输入 ID、密码就直接登录等。从本质上讲,它可以看作是你的身份证。但 cookie 不能作为代码执行,也不会传送病毒,且为你所专有,并只能由提供它的服务器来读取。保存的信息片段以“名/值”对(name-value pairs)的形式储存,一个“名/值”对仅仅是一条命名的数据。一个网站只能取得它放在你的计算机中的信息,它无法从其他的 cookie 文件中取得信息,也无法得到你的计算机上的其他任何东西。cookie 中的内容大多数经过了加密处理,因此一般用户看来只是一些毫无意义的字母数字组合,只有服务器的 CGI 处理程序才知道它们真正的含义。

10.4.1 cookie 工作原理

一般来说,cookie 通过 HTTP Headers 从服务器端返回到浏览器上。首先,服务器端在响应中利用 Set-Cookie header 来创建一个 cookie,然后,浏览器在它的请求中通过 cookie header 包含这个已经创建的 cookie,并且使它返回至服务器,从而完成浏览器的论证。

例如,我们创建了一个名字为 login 的 cookie 来包含访问者的信息,创建 cookie 时,服务器端的 Header 代码如下。

```
Set-Cookie:login=Mochael Jordan;path=/;domain=msn.com;
expires=Monday,01-Mar-99 00:00:01 GMT
```

上面这个 Header 会自动在浏览器端计算机的 cookie 文件中添加一条记录。浏览器将变量名为 login 的 cookie 赋值为 Michael Jordon。注意,在实际传递过程中,这个 cookie 的值是经过了 URLEncode 方法的 URL 编码操作的。这个含有 cookie 值的 HTTP Header 被保存到浏览器的 cookie 文件后,Header 就通知浏览器将 cookie 通过请求以忽略路径的方式返回到服务器,完成浏览器的认证操作。

客户端的每一项数据都是互相独立的,在需要读取时再分别获取。cookie 是将数据存储在客户端的技术之一。图 10-5 列出了 cookie 的使用过程。



图 10-5　cookie 的使用过程

10.4.2 cookie 和 session 的区别

cookie 和 session 完美地解决了无连接性质的 HTTP 协议,被广泛地应用在网站开发领域,用于保存用户信息、追踪个人和商业交易等。但是它们又有各自的优缺点和区别。

- cookie 数据存放在客户的浏览器上,session 数据存放在服务器上。
- cookie 不是很安全,别人可以分析存放在本地的 cookie 并进行 cookie 欺骗。考虑到安全,应当使用 session。
- session 会在一定时间内保存在服务器上。当访问增多,会比较影响服务器的性能,考虑到减轻服务器负担方面,应当使用 cookie。
- 单个 cookie 保存的数据不能超过 4KB,很多浏览器都限制一个站点最多保存 20 个 cookie。
- session 需要借助 cookie 才能正常地工作。如果客户端完全禁止 cookie,session 将失效。

10.5　cookie 的基本操作

cookie 在 PHP 网页的会话中起到了很重要的作用，在 PHP 中可以通过相关的 cookie 函数方便地操作 cookie，主要包括 cookie 的创建、访问和删除。

10.5.1　cookie 的创建

在 PHP 中可以使用 setcookie()函数，由于 cookie 是 HTTP 头标部分的内容，因此必须在输出任何数据之前调用 setcookie()函数，这个限制和 header()函数类似。setcookie()函数的语法格式为：

```
bool setcookie ( string name [ , string value [ , int expire [ , string path [ , string domain [
, int secure]]]]])
```

setcookie()函数的有关参数说明如下。

- name：这是必选参数，指定 cookie 的名称，使用 $_COOKIE['cookiename']调用名称为 cookiename 的 cookie。
- value：可选参数，表示 cookie 的值，保存在客户端，因此最好不要保存敏感或者机密的数据。当该参数的值为空字符串时，表示撤销客户端该 cookie 的资料。如果 name 是 cookiename，那么可以通过 $_COOKIE['cookiename']取值。
- expire：可选参数，表示 cookie 的有效截止时间，即过期时间或者有效时间，该参数必须是整型。例如，time() + 60 * 60 * 24 表示 cookie 在 24 小时后失效。如果没有设置，cookie 将会在结束会话后(一般是关闭浏览器)失效。
- path：表示存放 cookie 的有效路径，默认是当前目录。如果该参数设置为“/”，那么 cookie 在整个 domain 内有效；如果设置为/foo/，那么 cookie 只在 domain 的/foo/文件夹以及子文件夹内有效。
- domain：表示 cookie 有效的域名。例如，要使 cookie 能在 examle.com 域名小的所有子域都有效，该参数应该设置为.example.com。
- secure：表示 cookie 是否通过安全的 HTTPS 连接传送，默认为 false。当设置为 true 时，cookie 仅在安全的连接中被设置。

【示例 9】　使用 setcookie()函数创建 cookie。

eg9.php 代码如下。

```
<?php
  setcookie("username","Ken");
  //设置名为'username'的 cookie,值为'Ken'
  setcookie("password","123456",time()+24 * 60 * 60);
  //设置名为'password'的 cookie,值为'123456',有效期为 24 小时
  setcookie('sex','female',time()+60 * 60,"/foo/",".manager.com",1);
  //设置名为'sex',值为'female',有效期为 1 小时,
  //在域名 manager.com 中有效,在安全的连接中被设置
?>
```

【示例 10】 使用 setcookie()函数创建 cookie 数组。

eg10.php 代码如下。

```
<?php
  setcookie("info[name]","liubei",time()+24*60*60);
  setcookie("info[sex]","male",time()+24*60*60);
  setcookie("info[age]","23",time()+24*60*60);
?>
```

说明：

- $_COOKIE["info"]表示上面代码创建的全部 cookie。
- $_COOKIE["info"]["name"]、$_COOKIE["info"]["sex"]和 $_COOKIE["info"]["age"]则是数组的三个元素，值分别为 liubei、male 和 23。

10.5.2 cookie 的获取

当 cookie 设置后，可以通过 $_COOKIE["cookiename"]获得 cookie 的值。但是，cookie 变量只能在其他页面使用，在本页并不会生效，除非刷新本页。

【示例 11】 获取示例 9 和示例 10 创建的 cookie 变量和 cookie 数组。

eg11.php 代码如下。

```
<?php
  echo "<pre>";
  echo $_COOKIE['username']."\n";       //获得 cookie 变量，创建于示例 9
  echo $_COOKIE["info"]["name"]."\n";   //获得 cookie 数组的一个元素，创建于示例 10
  print_r($_COOKIE["info"]);            //获得 cookie 数组全部元素，创建于示例 10
?>
```

10.5.3 cookie 的删除

设置 cookie 有效期后，超过这个有效期 cookie 将自动删除。没有设置有效期的 cookie 在关闭浏览器后将自动删除。此外使用 PHP 中提供的 setcookie()函数可以删除 cookie。例如：

```
setcookie("username","");
```

上面的代码用于删除 cookie 变量 $_COOKIE["username"]。特别需要注意的是，在创建 cookie 时设置了特定的参数，在删除该 cookie 时也要提供这些参数，以便 PHP 正确地删除相应的 cookie 变量。

此外，在创建一个 cookie 变量时设置了它的有效期，若想删除它，还可以重新使用 setcookie()函数，在 setcookie()函数中把有效期设置为此时此刻之前的任一个时间即可。

【示例 12】 删除示例 9 和示例 10 创建的 cookie 变量和 cookie 数组。

eg12.php 代码如下。

```
<?php
  setcookie("username","");                  //删除$_COOKIE["username"];
  setcookie("password","123456",time()-1);   //删除$_COOKIE["password"];
```

```
    setcookie("info[name]","liubei",time()-1);    //删除数组元素$_COOKIE["info"]["name"]
    setcookie("info[sex]","male",time()-1);       //删除数组元素$_COOKIE["info"]["sex"]
    setcookie("info[age]","23",time()-1);         //删除数组元素$_COOKIE["info"]["age"]
    unset($_COOKIE["info"]);                      //作用与上面三个语句相同
?>
```

10.6　综合案例——使用 cookie 进行用户登录

使用 cookie 保存登录信息，便于下次再次登录。在家里登录某个管理系统网站，希望下次登录时不再输入用户名和密码。常常使用 cookie 来实现这一功能。

步骤 1　创建登录页面 login. php，该页面有文本框“用户名”、文本框“密码”、复选框“保留密码”和“提交”按钮等。需要输入“用户名”和“密码”并单击“登录”按钮，当“用户名”和“密码”都正确时即可登录到另外一页面 main. php。要求下次登录时保存“用户名”到“用户名”文本框。在“用户名”和“密码”都正确且选中了“保留密码”复选框的前提下，下次登录自动填写密码，否则需要输入密码。

创建 main. php 页面，判断存放在 cookie 中的用户名和密码是否正确，若正确则给出欢迎信息，否则强行跳转到登录页面 login. php。

登录页面 login. php 界面如图 10-6 所示。



图 10-6　登录页面 login. php 界面

eg13/login. php 代码如下。

```
<form name="form1" method="post" action="">
  用户名<input type="text" name="user" value="<?php if (isset($_COOKIE['user'])) echo $_
  COOKIE['user'];?>"  /><br />
  密  码<input type="password" name="pass" value="<?php if (isset($_COOKIE['pass'])) echo $_
  COOKIE['pass'] ;?>" /><br />
  <input type="checkbox" name="save" value="yes" <?php if (isset($_COOKIE['save'])) echo "
  checked";?>/>保留密码<br />
  <input type="submit" value="登录" name="button1" />
</form>
<?php
  header("content-type:text/html;charset=utf-8");
  ini_set("dislay_errors","0");
  if (isset($_POST['button1'])){
    if ($_POST['user']=='liubei' && $_POST['pass']=='123456'){
       setcookie('user','liubei',time()+30*24*60*60);
      //创建 cookie 变量 $_COOKIE['user']
```

```
            if ($_POST['save']=='yes'){
              setcookie('pass','123456',time()+30*24*60*60);
              //创建 cookie 变量 $_COOKIE['pass']
              setcookie('save','yes',time()+30*24*60*60);
              //创建 cookie 变量 $_COOKIE['save']
            }else{
              setcookie('pass','123456',time()-1);
              //销毁 cookie 变量 $_COOKIE['pass']
              setcookie('save','yes',time()-1);
              //销毁 cookie 变量 $_COOKIE['save']
            }
            echo "<script>window.location.href='main.php'</script>";
        }else{
          echo "<script>alert('用户名或密码错误!')</script>";
        }
      }
?>
```

步骤 2　编写 eg13/main.php 代码如下。

```
<?php
  if (isset($_COOKIE['user']) && isset($_COOKIE['pass'])){
    if ($_COOKIE['user']=='liubei' && $_COOKIE['pass']=='123456'){
      echo $_COOKIE['user'].",you are welcome.";
      //此处实现程序的其他功能
    }else
      echo "<script>window.location.href='login.php';</script>";
  }else
    echo "<script>window.location.href='login.php';</script>";
?>
```

说明：

(1) 第一次运行 login.php 页面，由于 $_COOKIE['user']和 $_COOKIE['pass']不存在，因此在用户名和密码文本框中不显示用户名和密码。

(2) 若某次运行 login.php 页面，在用户名和密码都正确的前提下保存“用户名”到 $_COOKIE['user']中(值为'liubei')，这样下次登录 login.php 页面时，因 $_COOKIE['user']存在，会在文本框中显示用户名(即在用户名文本框中显示 liubei)；

(3) 若某次运行 login.php 页面，在用户名和密码都输入正确的前提下且选中了“保留密码”复选框，此时不仅保存“用户名”到 $_COOKIE['user'](值为'liubei')，而且会把“密码”也保存到 $_COOKIE['pass'](值为'123456')，且会把状态“保留密码”也保存到 $_COOKIE['save']中(值为'yes')，方便下次登录。

(4) 在某次运行 login.php 页面时，若 $_COOKIE['user']存在，则显示用户名在用户名文本框中；若 $_CCOKIE['pass']存在，则显示密码在密码文本框中，此时一定会自动选中“保存密码”复选框。

(5) 特别强调的是：在用户名和密码都正确且不选中“保留密码”复选框时，才会清除保存在 cookie 中的密码，即清除 $_COOKIE['pass']，且会清除 $_COOKIE['save'](该 cookie 存在且值为'yes'，表示保存密码，不存在则表示不保存密码)。

(6) 在 main.php 页面中核对 $_COOKIE['user']和 $_COOKIE['pass']两个 cookie 变量：若二者都存在且值分别为'liubei'和'123456'，则给出欢迎信息，否则会强行跳转到 login.php 页面让用户登录。

(7) “value="<? php if (isset($_COOKIE['user'])) echo $_COOKIE['user'];? >"”语句表示如果 cookie 中保存有用户名信息，则把用户名显示在用户名文本框中。

(8) “value="<? php if (isset($_COOKIE['pass'])) echo $_COOKIE['pass'] ;? >"”语句表示如果 cookie 中保存有密码信息，则把密码显示在密码框中。

(9) “<? php if (isset($_COOKIE['save'])) echo "checked";? >”语句表示如果 cookie 中存在要保存的密码，则选中“保留密码”复选框。

步骤 3　运行程序，结果如图 10-7 和图 10-8 所示。



图 10-7　上一次正确输入用户名和密码且不选中“保留密码”复选框



图 10-8　上一次正确输入用户名和密码且选中“保留密码”复选框

10.7　习　　题

一、填空题

1. 设置 cookie 变量 city 的值为“武汉”，其实现代码为________。
2. 在 php.ini 文件中，设置________选项可以让程序自动启动会话。
3. 可以使用函数________来启动会话。
4. PHP 中使用________函数创建或删除 cookie。
5. session 和 cookie 更安全的是________。

二、选择题

1. PHP 提供的________函数用于启动会话。

   A. session_start()　　　　B. session_start_set()

   C. start_session()　　　　D. sessionstart()

2. 若当前时间是 2012 年 1 月 1 日，在下面的选项中，________选项指定的 cookie 会立

即消失。

A. setcookie("count","10",mktime(0,0,0,1,1,2017));

B. setcookie("count","10",mktime(11,0,0,11,11,2018));

C. setcookie("count","10",time() + 60 * 60 * 24));

D. setcookie("count","10",time-3600));

3. 执行下面的代码,运行结果是________。

```
<?php
  setcookie("citys[]","Changsha");
  setcookie("citys[hubei]","Wuhan");
  setcookie("citys[jiangxi]","Nanchang");
  setcookie("citys[]","Wuhan");
  setcookie("citys[]","Zhengzhou");
  $a = $_COOKIE["citys"];
  echo $a[0];
?>
```

A. Changsha　　B. Wuhan　　C. Nanchang　　D. Zhengzhou

三、程序设计题

创建一个登录程序,在页面中输入用户名和密码,并且选择 cookie 的有效时间,如果用户名和密码分别是 liubei 和 123456,则将输入的用户名和密码保存在 cookie 中,否则不保存用户名和密码。

第 11 章　文件和目录处理

知识点：

- 获取文件的属性
- 文件的基础操作
- 文件的高级操作
- 获取目录的属性
- 目录的基本操作

本章导读：

在实际网站开发过程中，用户常常需要对文件进行各种操作，例如访问目录、读写文件、上传下载文件等。幸好在 PHP 中提供了强大的文件与目录操作功能，这样用户就可以方便地进行文件和目录的各种操作了。

本章将详细地讲解文件属性的获取、文件的读写、文件的打开及关闭、文件的复制及删除、文件的上传及下载，文件夹的属性获取、文件夹的打开及关闭、文件夹的读取及删除和创建等操作。

11.1　获取文件的属性

操作文件时常常需要获取文件的类型、访问时间、修改时间以及文件的大小等，在 PHP 中提供了获取这些内容的相关函数，方便用户的调用。

11.1.1　文件的类型和大小

在文件夹中包含各种类型的文件，这些类型可以是 rar、txt、doc、gif、html、pdf、jpg、bmp 等，每个文件的大小也是不一样的。在 PHP 中使用 filetype()函数来获取文件的类型，使用 filesize()函数来获取文件的大小。

1. filetype()函数

filetype()函数用于取得文件的类型。基本形式如下。

```
string filetype ( string $filename )
```

filetype()函数返回 $filename 参数来指定文件的文件类型。其中 $filename 是包含路径在内的文件名。

返回值：返回文件的类型。可能的值有 fifo、char、dir、block、link、file 和 unknown。如果出错则返回 false。如果 stat 调用失败或者文件类型未知，filetype()还会产生一个 E_NOTICE 消息。返回值的含义如下。

✍ fifo：命名管道，常用于将信息从一个进程传递到下一个进程。

✍ char：字符设备，负责操作系统和设备之间的无缓冲数据交换。

✍ dir：目录，即文件夹。

✍ block：块设备，例如 CD_ROM。

✍ link：符号连接，指向文件的指针。

✍ file：硬链接，作为文件 inode 的指针。只要认为是一个文件，例如文档或可执行文件，都返回该类型。

✍ unknown：未知类型。

【示例 1】 使用函数 filetype()获取文件类型。

eg1. php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $file1 = "d:/book.JPG";
  $file2 = "plan.txt";
  $dir = "d:\phpStudy\WWW";
  echo "{$file1}的类型是".filetype($file1)."<br />";
  echo "{$file2}的类型是".filetype($file2)."<br />";
  echo "{$dir}的类型是".filetype($dir);
?>
```

程序运行结果如下。

```
d:/book.JPG 的类型是 file
plan.txt 的类型是 file
d:\phpStudy\WWW 的类型是 dir
```

2. filesize()函数

filesize()函数用于获取指定文件的大小。如果执行成功则返回文件大小的字节数，如果出错则返回 false，并且产生一个 E_WARNING 级别的错误。基本形式如下。

```
string filetype ( string $filename )
```

$filename 参数是包含路径在内的文件名。

返回值：返回文件大小的字节数，如果出错则返回 false 并生成一条 E_WARNING 级的错误。注意，因为 PHP 的整数类型是有符号整型而且很多平台使用 32 位整型，对 2GB 以上的文件，一些文件系统函数可能返回无法预期的结果。

【示例 2】 使用 filesize()函数获取文件的大小。

eg2. php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $file1 = "d:/book.jpg";
  $file2 = "plan.txt";
  echo "{$file1}的大小是".filesize($file1)."<br />";
  echo "{$file2}的大小是".filesize($file2);
?>
```

程序运行结果如下。

```
d:/book.jpg 的大小是 11226
plan.txt 的大小是 6
```

上述输出结果表示 D:/book.jpg 文件的大小是 11226 字节，plan.txt 文件的大小是 6 字节。

11.1.2　最后访问与修改时间

在现实的管理任务中确定文件的最后访问和修改时间也是非常重要的。在 PHP 中提供了几个函数用于这方面的操作。

1. fileatime()函数

fileatime()函数的基本形式为：

```
int fileatime ( string $filename )
```

函数返回文件上次被访问的时间，或者在失败时返回 false。时间以 UNIX 时间戳的方式返回。

【示例 3】 使用 fileatime()函数获取文件最后一次访问的时间。

eg3.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  ini_set('date.timezone','Asia/Shanghai');  //设置时区
  $filename = 'plan.txt' ;
  echo "{$filename}最后一次访问时间为".date("F d Y H:i:s.",fileatime($filename));
?>
```

程序运行结果如下。

```
plan.txt 最后一次访问时间为 October 25 2017 15:52:29.
```

说明：fileatime()函数返回的结果为时间戳(整型数)。如果使用不便，可以使用 date()函数格式化为用户方便识别的时间再来显示并查看。

2. filectime()函数

filectime()函数的基本形式为：

```
int filectime ( string $filename )
```

filectime()函数取得文件的 inode 修改时间，返回值是 UNIX 时间戳，是整型数。$filename 是包含路径在内的文件名。

返回值：返回文件上次 inode 被修改的时间，或者在失败时返回 false。时间以 UNIX 时间戳的方式返回。

【示例 4】 使用 filectime()函数获取文件最后一次修改的时间。

eg4.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
```

```
  ini_set('date.timezone','Asia/Shanghai');  //设置时区
  $filename = 'plan.txt' ;
  echo "{$filename}最后一次修改时间为".date("F d Y H:i:s.",filectime($filename));
?>
```

程序运行结果如下。

```
plan.txt 最后一次修改时间为 October 25 2017 15:52:29.
```

这里的最后改变是指定文件 filename 的 inode 最后改变时间，其中 inode(索引节点)用来存放档案及目录的基本信息，包含时间、档名、使用者及群组等，采用 UNIX 时间戳格式，有错误时返回 false。

3. filemtime()函数

filemtime()函数的基本形式如下。

```
int filemtime ( string $filename )
```

本函数返回文件中的数据块上次被写入的时间，即文件的内容上次被修改的时间。$filename 是包含路径在内的文件名。

返回文件上次被修改的时间，或者在失败时返回 false。时间以 UNIX 时间戳的方式返回。

【示例 5】 使用 filemtime()函数获取文件最后一次改变的时间。

eg5.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  ini_set('date.timezone','Asia/Shanghai');  //设置时区
  $filename = 'plan.txt' ;
  echo "{$filename}最后一次修改时间为".date("F d Y H:i:s.",filemtime($filename));
?>
```

程序运行结果如下。

```
plan.txt 最后一次修改时间为 October 25 2017 15:52:38.
```

最后修改指的是文件的内容改变，采用 UNIX 时间戳格式，有错误时返回 false。

11.1.3 其他属性

除了前面介绍的常用与文件有关的函数之外，还可以通过其他函数获取文件的相关信息。具体说明如表 11-1 所示。

表 11-1 获取文件其他属性的函数

函数名称	说明
fileinode()	获取文件的 inode 编号，出错时返回 false
filegroup()	取得该文件所属组的 ID，以数字格式返回，用 posix_getgrgid()函数将其解析为组名，出错时返回 false，并产生一个 E_WARNING 级别的错误
fileowner()	返回文件所有的用户 ID，用户 ID 以数字形式返回，可以使用 posix_getpwuid()将其解析为用户名。出错时返回 false

续表

函 数 名 称	说　　明
fileperms()	返回文件的访问权限，出错则返回 false
is_executable()	判断文件是否是可执行文件，如果文件存在且可执行则返回 true
is_readable()	判断文件是否可读。存在且可读则返回 true
is_writeable()	判断文件是否可写。存在且可写则返回 true
file_exists()	判断文件是否存在，存在则返回 true
is_file()	判断是否是文件，是文件则返回 true
is_dir()	判断是否为文件夹(目录)，是文件夹则返回 true

【示例 6】 获取文件信息。

eg6.php 代码如下。

```
<?php
  function perms_str($perms){
    if (($perms & 0xC000) == 0xC000) {
        // Socket
        $info = 's';
    } elseif (($perms & 0xA000) == 0xA000) {
        // Symbolic Link
        $info = 'l';
    } elseif (($perms & 0x8000) == 0x8000) {
        // Regular
        $info = '-';
    } elseif (($perms & 0x6000) == 0x6000) {
        // Block special
        $info = 'b';
    } elseif (($perms & 0x4000) == 0x4000) {
        // Directory
        $info = 'd';
    } elseif (($perms & 0x2000) == 0x2000) {
        // Character special
        $info = 'c';
    } elseif (($perms & 0x1000) == 0x1000) {
        // FIFO pipe
        $info = 'p';
    } else {
        // Unknown
        $info = 'u';
    }
    // Owner
    $info .= (($perms & 0x0100) ? 'r' : '-');
    $info .= (($perms & 0x0080) ? 'w' : '-');
    $info .= (($perms & 0x0040) ?
                (($perms & 0x0800) ? 's' : 'x' ) :
                (($perms & 0x0800) ? 'S' : '-'));
    // Group
    $info .= (($perms & 0x0020) ? 'r' : '-');
```

```
    $info .= (($perms & 0x0010) ? 'w' : '-');
    $info .= (($perms & 0x0008) ?
                (($perms & 0x0400) ? 's' : 'x' ) :
                (($perms & 0x0400) ? 'S' : '-'));
    // World
    $info .= (($perms & 0x0004) ? 'r' : '-');
    $info .= (($perms & 0x0002) ? 'w' : '-');
    $info .= (($perms & 0x0001) ?
                (($perms & 0x0200) ? 't' : 'x' ) :
                (($perms & 0x0200) ? 'T' : '-'));
    return $info;
  }
  $file = 'plan.txt';
  if (file_exists($file))
     $perms = fileperms($file);
  echo perms_str($perms);
?>
```

程序运行结果如下。

```
-rw-rw-rw-
```

11.2 文件的基本操作

程序员经常需要访问文件夹和文件，以便收集信息对文件系统进行必要的操作。例如，打开、读写、复制及删除文件。本节将讲述使用 PHP 函数对文件进行基本的操作。

11.2.1 文件的打开

在进行文件操作之前，通常需要建立与文件资源的连接。同样，结束该资源的操作后，应当关闭连接。

在 PHP 中使用 fopen()函数来进行打开文件或者 URL 的操作。基本形式如下。

```
resource fopen ( string $filename , string $mode [, bool $use_include_path = false [,
resource $context ]] )
```

fopen()将 filename 指定的名字资源绑定到一个流上，如果 filename 是"scheme://..." 的格式，则被当成一个 URL，PHP 将搜索协议处理器(也被称为封装协议)来处理此模式；如果该协议尚未注册封装协议，PHP 将发出一条消息来帮助检查脚本中潜在的问题并将 filename 当成一个普通的文件名继续执行下去。

如果 PHP 认为 filename 指定的是一个本地文件，将尝试在该文件上打开一个流。该文件必须是 PHP 可以访问的，因此需要确认文件访问权限允许该访问。如果激活了安全模式或者 open_basedir，则会应用进一步的限制。

如果 PHP 认为 filename 指定的是一个已注册的协议，而该协议被注册为一个网络 URL，PHP 将检查并确认 allow_url_fopen 已被激活。如果关闭了，PHP 将发出一个警告，

而 fopen 的调用则失败。

mode 参数可以指定的模式见表 11-2。

表 11-2　mode 参数可以指定的模式

mode 取值	说　　明
r	文件以只读方式打开，将文件指针指向文件头
r+	文件以读写方式打开，将文件指针指向文件头
w	文件以写入方式打开，将文件指针指向文件头并将文件大小截为零。如果文件不存在，则尝试创建
w+	文件以读写方式打开，将文件指针指向文件头并将文件大小截为零。如果文件不存在，则尝试创建
a	文件以写入方式打开，将文件指针指向文件末尾。如果文件不存在，则尝试创建
a+	文件以读写方式打开，将文件指针指向文件末尾。如果文件不存在，则尝试创建
x	创建文件并以写入方式打开，将文件指针指向文件头。如果文件已存在，则 fopen()调用失败并返回 false，并生成一条 E_WARNING 级别的错误信息。如果文件不存在，则尝试创建。这和给底层的 open(2)系统调用指定 O_EXCL|O_CREAT 标记是等价的
x+	创建文件并以读写方式打开，其他的行为和 x 值的效果一样

【示例 7】 使用 fopen()函数以不同形式打开文件。

eg7.php 代码如下。

```
<?php
  $file1 = fopen('plan.txt','r');                                  //以只读方式打开
  $file2 = fopen('d:/plan2.txt','r+');                             //以读写方式打开
  $file3 = fopen('d:/phpstudy/www/a.txt','w');                     //以写入方式打开
  $file4 = fopen('http://www.163.com','r');                        //打开网址
  $file5 = fopen('fpt://user:password@itzcn.com/log.txt','w');     //写入方式
?>
```

11.2.2　文件的关闭

一旦完成资源的处理，就要撤销其指针。fclose()函数用于这样的操作，关闭之前打开的 handle 指定文件指针。基本形式如下。

```
bool fclose ( resource $handle )
```

fclose()函数关闭一个已打开的文件指针。$handle 是文件指针，文件指针必须有效，并且是通过 fopen()或 fsockopen()成功打开的。

返回值：成功时返回 true，失败时返回 false。

【示例 8】 使用 fclose()函数关闭文件。

eg8.php 代码如下。

```
<?php
  $file = 'plan.txt';
  if (file_exists($file)){
    $handle = fopen($file ,'r');
    //其他代码
```

```
        fclose($handle);
    }
?>
```

11.2.3　文件的读取

打开文件后，可以使用PHP提供的函数进行读取等操作。这些函数不仅可以读取一个字符，还可以一次性读取整个文件。

在PHP中提供了8个与读取文件有关的函数，如表11-3所示。

表11-3　PHP提供的与读取文件有关的函数

函数名称	说　　明
file()	把文件读入到数组，元素之间以换行符分隔，换行符附加在每个元素末尾
file_get_contents()	把文件读入到字符串中
fread()	读取已经打开的文件，并且可以规定读取几个字符
fgetc()	从打开的文件中读取字符，并且返回一个字符
fgets()	从打开的文件中读取一行
fgetss()	与fgets()相似，不同的是它会自动过滤HTML和PHP标记
fgetcsv()	从文件指针中读取一行，并且解析CSV字段，然后再返回一个包含这些字段的数组
readfile()	读取一个文件并且写入缓冲区，如果该函数执行成功，则返回从文件中读入的字节数；执行失败则返回false

下面介绍其中的几个函数。

1. file()函数

file()函数是读取文件时经常使用的一个函数，它把整个文件读入到一个数组中，各个元素之间以换行符分隔，同时换行符附加在每个元素末尾。基本形式如下。

```
array file ( string $filename [, int $flags = 0 [, resource $context ]] )
```

$filename是文件的路径。可选参数$flags可以是以下一个或多个常量。

- FILE_USE_INCLUDE_PATH：在include_path中查找文件。
- FILE_IGNORE_NEW_LINES：在数组每个元素的末尾不要添加换行符。
- FILE_SKIP_EMPTY_LINES：跳过空行。

可选参数$context用于指定文件句柄环境。

返回值：将文件以数组的形式返回，每个元素包含一个换行符，换行符附加在每个元素后，执行失败则返回false。

【示例9】　使用file()函数读文件。

eg9.php代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $file = "eg9.txt";
  if (file_exists($file)){
    $lines = file($file);
```

```
    foreach($lines as $line_num=>$line)
      echo nl2br($line) ;                //自带换行符,转换为<br />
  }else
    "文件不存在";
?>
```

程序运行结果如图 11-1 所示,显示的文字为文件中的内容。

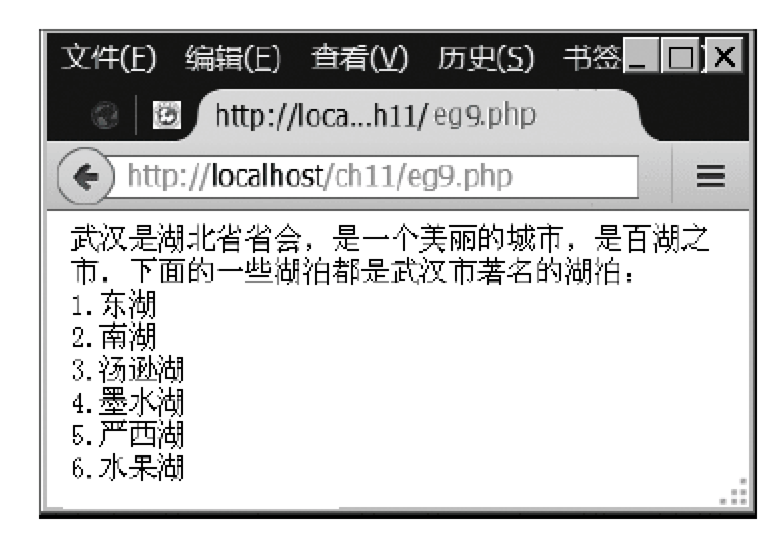


图 11-1　使用 file()函数读取文件的内容

2. file_get_contents()函数

file_get_contents()函数用于读取文件。与 file()函数不同的是,它把一个文件读入到一个字符串中。基本格式如下。

```
string file_get_contents ( string $filename [, bool $use_include_path = false
[, resource $context [, int $offset = -1 [, int $maxlen ]]]] )
```

file_get_contents()函数将在 $offset 参数所指定的位置开始读取长度为 $maxlen 的内容。如果失败,将返回 false。file_get_contents()函数是用来将文件的内容读入到一个字符串中的首选方法。如果操作系统支持,还会使用内存映射技术来增强性能。

【示例 10】 使用 file_get_contents()函数读文件。

eg10.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $file = "eg9.txt";
  if (file_exists($file)){
    $str = file_get_contents($file);
    echo nl2br($str);
  }else
    "文件不存在";
?>
```

程序运行结果和 eg9.php 相同。

3. fread()函数

fread()函数用于读取已经打开的文件,可以指定读取的字节数。基本形式如下。

```
string fread ( resource $handle , int $length )
```

fread()函数从文件指针 $handle 处开始读取最多 $length 个字节。当函数读完 $length 个字节后,或者到达文件为 EOF 或者一个包可以用时,就停止读取该文件。

【示例 11】 使用 fread()函数读文件。

eg11. php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $file = "eg9.txt";
  if (file_exists($file)){
    $handle = fopen($file,'r');
    //$str = fread($handle,21);  //读取 7 个汉字。utf8 编码中一个汉字等于 3 个字符
    $str = fread($handle,filesize($file));  //读出全部字符。filesize 用于求文件字节数
    echo nl2br($str);
  }else
    "文件不存在";
?>
```

程序运行效果与 eg9. php 相同。

4. fgets()函数

fgets()函数用于从文件指针中读取一行。基本形式为:

```
string fgets ( resource $handle [, int $length ] )
```

该函数用于从文件中读取一行。$handle 是文件指针。文件指针必须是有效的,必须指向由 fopen()函数或 fsockopen()函数成功打开的文件(并且未由 fclose()函数关闭)。$length 表示从 $handle 指向的文件中读取一行并返回长度最多为 1 字节的字符串。碰到换行符(包括在返回值中)、EOF 或者已经读取了 1 字节后停止(看先碰到那一种情况)。如果没有指定 length,则默认为 1KB,或者说 1024 字节。

返回值:从指针 $handle 指向的文件中读取了 $length 的 1 字节后返回字符串。如果文件指针中没有更多的数据,则返回 false。错误发生时返回 false。

【示例 12】 使用 fgets()函数读文件。

eg12. php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $handle  = @fopen("eg9.txt","r");
  if ($handle){
    while (($buffer = fgets($handle,4096)) !== false){
      echo $buffer;
    }
    if (!feof($handle)){
      echo "Error: unexpected fgets() fail\n" ;
    }
    fclose($handle);
  }
?>
```

程序运行结果与 eg9. php 相同。

11.2.4 文件的写入

如果想写入文件,在 PHP 中可以使用 fwrite()、fputs()和 file_put_contents()函数来

实现。

1. fwrite()函数

fwrite()函数用于写入文件，如果写入成功则返回写入的字节数，出现错误则返回false。fwrite()函数的基本形式如下。

```
int fwrite ( resource $handle , string $string [, int $length ] )
```

fwrite()写入文件(可安全用于二进制文件)。fwrite()把$string的内容写入文件指针$handle处。$handle是文件系统指针，是典型地由fopen()创建的resource(资源)。$string是用于写入的字符串。$length是写入的长度，如果指定了$length，当写入了$length个字节或者写完了$string以后，写入就会停止，看先碰到哪种情况。注意如果给出了$length参数，则magic_quotes_runtime配置选项将被忽略，而$string中的斜线将不会被抽去。

返回值：fwrite()函数返回写入的字符数，出现错误时则返回false。

【示例13】 使用fwrite()函数写文件。

eg13.php代码如下。

```
<?php
  $fp = fopen ( 'data.txt' , 'w' );     //以写入方式打开
  fwrite ( $fp ,  '1' );
  $length = fwrite ( $fp ,  '23' );     //写入字符串'23',并返回字节数 2
  fclose ( $fp );
  echo $length;
?>
```

运行程序，文件data.txt若不存在，则新建该文件，并写入1和23，因此文件内容变成123；若文件data.txt存在，则打开它，在写入内容时会清除原来的内容，因此文件内容也会变成123。此外输出的$length为2，即写入的字节数。

【示例14】 使用fwrite()函数写文件，添加模式。

eg14.php代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  $filename = 'test.txt' ;
  $somecontent = "添加这些文字到文件\n" ;
  //首先我们要确定文件存在并且可写
  if (is_writable( $filename )) {
    /* 在这个例子里,我们将使用添加模式打开$filename,因此,文件指针将会在文件的末尾,那
       就是当我们使用 fwrite()时,$somecontent 将要写入的地方 */
    if (!$handle  =  fopen ( $filename ,  'a' )) {
      echo  "不能打开文件  $filename " ;
      exit;
    }
    //将$somecontent 写入我们打开的文件中。
    if (fwrite($handle,$somecontent)===false){
      echo "不能写入文件{$filename}";
      exit;
```

```
    }
    echo  "成功地将  $somecontent  写入文件 $filename " ;
    fclose ( $handle );
  }else{
    echo "文件{$filename}不可写";
  }
?>
```

运行程序：首先要新建文件 test. txt，再运行文件，成功将字符串"添加这些文字到文件\n"写入文件 test. txt。再次运行会写入字符串两次，即以添加方式写入文件。

fputs()是 fwite()的别名。

2. file_put_contents()函数

file_put_contents()函数和依次调用 fopen()函数、fwrite()函数和 fclose()函数的效果是一样的，用于将一个字符串写入文件。基本形式为：

```
int file_put_contents ( string $filename , mixed $data [, int $flags =0 [, resource $context
]] )
```

如果文件不存在则创建该文件，如果文件存在则覆盖该文件，除非把 flag 设置为 FILE_APPEND 模式。

有关参数说明如下。

(1) $filename 是要写入数据的文件名；

(2) $data 是要写入的数据。类型可以是 string、array 或者是 stream 资源(如上面所说的那样)。如果 data 指定为 stream 资源，这里 stream 中所保存的缓存数据将被写入指定文件中，这种用法就类似于使用 stream_copy_to_stream()函数。参数 data 可以是数组(但不能为多维数组)，这就相当于 file_put_contents($filename, join('', $array))。

(3) $flags 的值可以是以下 flag 使用 OR (|) 运算符进行的组合。

① FILE_USE_INCLUDE_PATH：在 include 目录里搜索 filename。更多信息可参见 include_path。

② FILE_APPEND：如果文件 filename 已经存在，则追加数据而不是覆盖。

③ LOCK_EX：在写入时获得一个独占锁。

(4) context：一个 context 资源。

返回值：该函数将返回写入文件内数据的字节数，失败时返回 false。

【示例 15】 使用 file_put_contents()函数写文件。

eg15. php 代码如下。

```
<?php
  $file = 'people.txt';
  //打开文件并获取现有内容
  $current = file_get_contents($file);
  /* 第一次运行产生警告信息: file_get_contents(people.txt): failed to open stream: No such
     file or directory in D:\phpStudy\WWW\ch11\eg15.php on line 4,但是仍然建立了文件
     people.txt。第二次以及以后,每次运行都不会有警告 */
  //将一个新的人名添加到文件中
  $current .="John Smith\n" ;
```

```
  //将内容写回文件
  file_put_contents($file,$current);
?>
```

【示例 16】　使用 file_put_contents()函数写文件,并使用 flag 参数。

eg16.php 代码如下。

```
<?php
  $file = 'people.txt';
  $person = "John Smith\n" ;
  file_put_contents($file,$person, FILE_APPEND | LOCK_EX );
  //即使 people.txt 文件不存在也不会产生警告信息,而是直接创建文件,并以添加方式写入内容
?>
```

11.2.5　文件的复制

在 PHP 中使用 copy()函数实现文件的复制功能。如果执行成功则返回 true,执行失败则返回 false。基本形式如下。

```
bool copy ( string $source , string $dest [, resource $context ] )
```

copy()函数用于复制文件。将文件从 $source 复制到 $dest。如果要移动文件,请使用 rename()函数。$source 是源文件路径,$dest 是目标文件路径。如果 $dest 是一个 URL,则如果封装协议不支持覆盖已有的文件时,复制操作会失败。$context 是一个合法的使用 stream_context_create()建立的文件资源。

返回值:成功时返回 true,或者在失败时返回 false。

【示例 17】　使用 copy()函数复制文件。

eg17.php 代码如下。

```
<?php
  $file = 'eg17.txt' ;
  $newfile = 'eg17bak.txt';
  if (!copy($file,$newfile)){
    echo  "failed to copy  $file ...\n" ;
  }
?>
```

11.2.6　文件的删除

在 PHP 中使用 unlink()函数删除文件,删除成功则返回 true,删除失败则返回 false。基本形式如下。

```
bool unlink ( string $filename [, resource $context ] )
```

unlink()函数用于删除文件。与 UNIX C 的 unlink()函数相似。发生错误时会产生一个 E_WARNING 级别的错误。$filename 是要删除的文件。

返回值:成功时返回 true,或者在失败时返回 false。

【示例 18】　使用 unlink()函数删除文件。

eg18. php 代码如下。

```
<?php
  $fh = fopen('eg18.html','a');                 //创建文件 eg18.html
  fwrite($fh,'<h1>Hello world!</h1>');          //写入文件
  fclose ($fh);                                 //关闭文件
  unlink ('eg18.html');                         //删除文件 eg18.html
?>
```

11.3 非线性读写文件

非线性文件处理是指对文件的内容进行跳跃式的访问。例如,用户对文件的内容不是顺序地读取,也不是顺序地写入。这就需要使用非线性读取或写入的函数了。

11.3.1 fseek()函数

fseek()函数用于在打开的文件中定位,执行成功则返回 0,失败则返回-1。使用 fseek()函数到达新的位置后,可以使用 fgets()函数、fscanf()函数,或在其他任何地方再读取数据。fseek()函数的基本形式如下。

```
int fseek ( resource $handle , int $offset [, int $whence = SEEK_SET ] )
```

fseek()函数在与 $handle 关联的文件中设置文件指针的位置。新位置从文件头开始,以字节数度量,即以 whence 指定的位置再加 offset。whence 参数是一个可选参数,它的值说明如下。

- SEEK_SET：设定位置等于 offset 字节。
- SEEK_CUR：设定位置为当前位置加上 offset。
- SEEK_END：设定位置为文件尾加上 offset。

返回值：成功则返回 0,否则返回-1。注意移动到 EOF 之后的位置不算错误。

【示例 19】 使用 fseek()函数定位文件。

eg19. php 代码如下。

```
<?php
  $fp = fopen('eg19.txt','r');
  $data = fgets($fp);
  $result = fseek($fp,-10,SEEK_CUR);            //向后面跳转 10 个字符
?>
```

11.3.2 ftell()函数

ftell()返回文件指针读/写的位置。基本形式为：

```
int ftell ( resource $handle )
```

参数 $handle 是文件指针。文件指针必须是有效的,且必须指向一个通过 fopen()函

数或 popen()函数成功打开的文件。在附加模式(加参数 "a" 打开文件)中,ftell()会返回未定义错误。

返回值:返回由 $ handle 指定的文件指针的位置,也就是文件流中的偏移量,如果出错则返回 false。

【示例 20】 使用 fseek()函数定位文件。

eg20. php 代码如下。

```
<?php
  // 打开文件并读数据
  $fp = fopen("eg20.txt","r");
  $data = fgets($fp,12);                    //读取 11 个字符
  // 文件流中的偏移量
  echo ftell($fp)."<br />";
  fclose($fp);
  echo $data;
?>
```

11.3.3 rewind()函数

rewind()函数用于倒回指针位置,将文件句柄位置指针设置为文件流的开头。基本形式如下。

```
bool rewind ( resource $handle )
```

将 $ handle 的文件位置指针设为文件流的开头。如果将文件以附加(a 或者 a+)模式打开,写入文件的任何数据总是会被附加在后面,不管文件指针的位置。参数 $ handle 是合法的文件指针,并且指向由 fopen()成功打开的文件。

返回值:成功时返回 true,或者在失败时返回 false。

【示例 21】 rewind()函数的使用。

eg21. php 代码如下。

```
<?php
  $handle = fopen('eg21.txt','r+');  //以写入模式打开文件
  fwrite($handle,'Really long sentence.');
  rewind($handle);
  fwrite($handle,'Foo');
  rewind($handle);
  echo fread($handle,filesize('eg21.txt'));
  fclose($handle);
?>
```

文件 eg21. txt 的初始内容为 1234567890abcdefg。运行本程序后,文件 eg21. txt 的内容为“Foolly long sentence. ”。

【示例 22】 rewind()函数的使用。

eg22. php 代码如下。

```
<?php
```

```
    $handle = fopen('eg22.txt','a+');   //以添加模式打开文件
    fwrite($handle,'Really long sentence.');
    rewind($handle);
    fwrite($handle,'Foo');
    rewind($handle);
    echo fread($handle,filesize('eg22.txt'));
    fclose($handle);
?>
```

文件 eg22.txt 的初始内容为 1234567890abcdefg。运行本程序后文件 eg22.txt 的内容变为“1234567890abcdefgReally long sentence.Foo”。

11.4 文件的高级操作

在网站上经常需要对文件进行上传和下载操作，在 PHP 中也不例外。在 PHP 中要想实现文件的上传和下载等操作，有时候还需要进行适当的配置。

11.4.1 文件的上传

在 PHP 中实现文件的上传可以使用相关的函数和指令。本节将分别介绍。

1. 资源指令

在用 PHP 实现文件上传时有一些配置指令可以用于上传的精细调节。比如是否启动 PHP 上传，可允许上传文件的大小，可允许的最大脚本内存分配和其他各种重要资源基准。表 11-4 列举出了 PHP 中文件上传时的资源指令。

表 11-4 PHP 提供的与读取文件有关的函数

指令名称	类型	作用域	默认值	说明
file_uploads	boolean	PHP_INI_SYSTEM	1	用于设置服务器上的 PHP 脚本是否接受文件上传
max_execution_time	integer	PHP_INI_ALL	30	用于确定 PHP 脚本在注册一个致命错误之前可以执行的最长时间，单位为秒
memory_limit	integer	PHP_INI_ALL	8MB	用于设置脚本可以分配的最大内存，单位为 MB，整数值。这可以避免失控的脚本独占服务器内存。它只有在编译时设置了 enable_memory_limit 的情况下才有效
upload_max_filesize	integer	PHP_INI_SYSTEM	2MB	确定允许最大的上传文件，单位是 MB。必须小于 post_max_size 的值，因为它只应用于通过 file 输入类型传递的信息。注意，整数值后面必须加 M
upload_tmp_dir	string	PHP_INI_SYSTEM	NULL	上传的文件在处理之前必须成功地传输到服务器，因此必须指定一个位置用于临时存放这些文件，直到文件最终移动到目的地为止

续表

指令名称	类型	作　用　域	默认值	说　　明
post_max_size	integer	PHP_INI_SYSTEM	8MB	确定通过 POST 方法可以接收信息的最大值，单位为 MB。其值应该大于 upload_max_filesize 的值，因为除了上传的文件之外，还可以有其他上传的表单域。整数值后面一定要加 M

2. $_FILE 数组

$_FILES 是一个全局数组，用于存储各种与文件上传的信息。这些信息对于通过 PHP 脚本把文件上传到服务器非常重要。$_FILES 是一个二维数组。具体说明如下。

(1) $_FILES["file"]["name"]：被上传文件的名称。

(2) $_FILES["file"]["type"]：被上传文件的类型。

(3) $_FILES["file"]["size"]：被上传文件的大小，以字节计。

(4) $_FILES["file"]["tmp_name"]：存储在服务器上的文件的临时副本的名称。

(5) $_FILES["file"]["error"]：由文件上传导致的错误代码。共有 5 个返回值，其中 0 表示成功，其他表示失败，具体说明如下。

① 0：没有错误发生，文件上传成功，返回结果是 UPLOAD_ERR_OK。

② 1：上传的文件超过了 php.ini 中 upload_max_filesize（默认情况为 2MB）选项限制的值，返回结果是 UPLOAD_ERR_INI_SIZE。

③ 2：上传文件的大小超过了 HTML 表单中 MAX_FILE_SIZE 选项指定的值，返回结果是 UPLOAD_ERR_FORM_SIZE。

④ 3：文件只有部分被上传，返回结果是 UPLOAD_ERR_PARTIAL。

⑤ 4：没有文件被上传，返回结果是 UPLOAD_ERR_NO_FILE。

⑥ 5：上传文件大小为 0。

3. 与上传相关的函数

在 PHP 中提供了两个与文件上传相关的函数，它们是 is_uploaded_flie()函数和 move_uploaded_file()函数。基本形式为：

```
bool is_uploaded_file ( string $filename )
```

is_uploaded_file()函数用于判断文件是否是通过 HTTP POST 上传的。如果 filename 所给出的文件是通过 HTTP POST 上传的，则返回 true。这可以用来确保恶意的用户无法欺骗脚本去访问本不能访问的文件，例如 /etc/passwd。参数 $filename 是要检查的文件名。

返回值：成功时返回 true，或者在失败时返回 false。

move_uploaded_file()函数将上传的文件移动到新位置。基本形式为：

```
bool move_uploaded_file ( string $filename , string $destination )
```

本函数检查并确保由 $filename 指定的文件是合法的上传文件（即通过 PHP 的 HTTP POST 上传机制所上传的）。如果文件合法，则将其移动为由 $destination 指定的文件。这种检查显得格外重要，如果上传的文件有可能会造成对用户或本系统的其他用户显

示其内容。参数 $filename 是要上传的文件的文件名。$destination 是要将文件移动到的位置。

返回值：成功时返回 true。如果 $filename 不是合法的上传文件，不会出现任何操作，move_uploaded_file()将返回 false。如果 $filename 是合法的上传文件，但出于某些原因无法移动，不会出现任何操作，move_uploaded_file()函数将返回 false，此外还会发出一条警告。

4. 文件上传举例

下面的程序将图片上传并显示出来。

【示例 23】 新建文件夹 eg23，在该文件夹中新建文件夹 img。在 eg23 文件夹新建网页 index.php，index.php 通过表单提交上传图片文件到 img 文件夹并把图片显示在网页中。本程序需要添加表单、“提交”按钮。

eg23/index.php 代码如下。

```
<form action="" method="post" enctype="multipart/form-data" name="form1">
  上传图片<input type="file" name="fileField1" />
  <input type="submit" name="button" value="提交">
</form>
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  if (isset($_POST['button'])){        //单击了“提交”按钮
    if (($errortype =$_FILES['fileField1']['error'])>0){   //上传出错
      switch($errortype){
        case 1:$errorinfo='上传文件超过了 upload_max_filesize 限定的值';break;
        case 2:$errorinfo='上传文件超过了 MAX_FILE_SIZE 限定的值';break;
        case 3:$errorinfo='文件只上传了一部分';break;
        case 4:$errorinfo='没有指定上传文件就提交了!';break;
      }
      die($errorinfo);
    }
    //程序能执行到这里,说明$_FILES['fileField1']['error']的值为 0
    //$_FILES['fileField1']['error']值为 0,表示文件上传成功,则执行下面的代码
    $pictype =array('image/jpeg','image/pjpeg','image/png','image/gif');  //图片类型
    if (in_array($_FILES['fileField1']['type'],$pictype)){     //正确的图片格式
      echo "文件上传成功!\n";  //图片文件上传成功,但暂时保存在临时位置
      echo "上传的文件名为:".$_FILES['fileField1']['name']."\n";
      echo "上传文件类型为:".$_FILES['fileField1']['type']."\n";
      echo "上传文件大小为:".$_FILES['fileField1']['size']."\n";
      echo "上传文件临时副本名称为:".$_FILES['fileField1']['tmp_name']."\n";
      if (file_exists("img/".$_FILES['fileField1']['name'])){  //目标位置文件存在
        echo "你上传的文件{$_FILES['fileField1']['name']}已经存在,请换一个文件\n";
        //此时不将上传的临时副本移动到目标位置
      }else{
        $src =$_FILES['fileField1']['tmp_name'];         //上传文件的临时副本
        $des ="img/".$_FILES['fileField1']['name'];      //上传文件要移动的目标位置
        move_uploaded_file($src,$des);
        //将上传的文件(临时副本)移动到目标位置"img/".$_FILES['fileField1']['name']
        echo "上传成功,保存在"."img/".$_FILES['fileField1']['name']."图片显示如下。\n";
        $pic ="img/".$_FILES['fileField1']['name'];
```

```
            echo "<img src=$pic />";
          }
        }else{
          echo "请选择 gif、png 或 jpg 格式的图片文件。";
        }
      }
    ?>
```

运行程序，结果如图 11-2～图 11-4 所示。



图 11-2　第一次运行并选择上传文件为 php. gif



图 11-3　上传 php. gif 并提交后的结果

说明：

- 第一次选择文件 php. gif 上传，上传成功，上传后的临时副本为 C:\Documents and Settings\hdh\Local Settings\Temp\phpA3. tmp。此时，目标位置 img 不存在文件 php. gif，于是把临时副本 C:\Documents and Settings\hdh\Local Settings\Temp\phpA3. tmp 移动到目标位置 img，并改名为 php. gif(与原来的文件同名)。
- 第二次再选择文件 php. gif 上传，仍然上传成功，上传后的临时副本为 C:\Documents and Settings\hdh\Local Settings\Temp\phpA6. tmp(与第一次的临时



图 11-4　第二次再选择上传 php. gif 并提交后的结果

副本不一定会同名)。由于此时目标位置 img 存在文件 php. gif,本程序不会移动临时副本到目标位置。

- 第一次或第二次都会显示文件 php. gif。
- 若上传文件出错,则程序会停止在"die($ errorinfo);"语句,程序不会执行后面的代码。
- 若选择上传其他类型的文件,本程序会提示:"请选择 gif、png 或 jpg 格式的图片文件"。

11.4.2　文件的下载

大多数情况下,需要把用户上传的文件供用户下载查看,在这种情况下,就可以用到 PHP 的文件下载技术。PHP 实现文件下载时,首先需要判断用户是否单击了下载链接。代码如下。

```
<a href="?action=download">下载</a>
```

然后指定文件的名称和路径,并且打开该文件。

```
header("content-type:application/octet-stream");
header("Accept-Ranges:bytes");
header("Accept-Length:".filesize('1.jpg'));
header("Content-Disposition:attachment;filename=1.jpg");
```

最后使用 fread()函数将文件内容直接从网页输出,让浏览器提示用户下载。所有的这些处理都在服务器上进行,因此用户是不会知道文件的具体位置信息的,是非常安全的一种下载方式。

【示例 24】 使用 header 实现文件的下载,先新建文件夹 eg24,再新建 index. html 和 eg24. php。

eg24/index. html 代码如下。

```
<a href="eg24.php">down</a>
eg24/eg24.php 代码如下。
<?php
```

```
  $filename = 'pdf/eg24.pdf';                //服务器上供下载的文件
  header('Content-type: application/pdf');   //文件的类型
  header('Content-Disposition: attachment; filename="eg24new.pdf"');
  //下载后的文件名
  readfile("$filename");
  exit();
?>
```

说明：

- eg24/index.html 主要是提供超链接供用户单击下载。而 eg24/eg24.php 则实现下载功能。
- $filename = 'pdf/eg24.pdf'语句指定了要下载的文件，在服务器上，用户看不到文件名和路径，相对于 HTML 中通过超链接和路径下载而言更安全。
- header('Content-type：application/pdf')语句指明了要下载的文件的类型。
- header('Content-Disposition：attachment；filename="eg24new.pdf"')语句用于实现下载，eg24new.pdf 是下载后的文件名。
- readfile("$filename")语句用于从页面中输出内容，是必不可少的。

11.5　获取目录属性

文件可以直接放置在磁盘下，如果文件比较多，则需要将文件进行分类，并且要将同一类的文件保存在一起，比如同一个目录（文件夹）中。本节将介绍常用的几个获得目录属性的函数。

11.5.1　解析文件的路径

将目录路径解析为各个属性非常有用，这样的函数主要有以下 3 个。

1．basename()函数

basename()函数用于返回路径中的文件名部分。基本形式如下。

```
string basename ( string $path [, string $suffix ] )
```

给出一个包含有指向一个文件的全路径的字符串，本函数返回基本的文件名。参数 $path 是一个路径。在 Windows 中，斜线(/)和反斜线(\)都可以用作目录分隔符。在其他环境下是斜线(/)。另外，如果文件名是以 $suffix 结束的，这一部分也会被去掉。

返回值：返回 $path 基本的文件名。

【示例 25】 使用 basename()函数获得路径中的文件名部分。

eg25.php 代码如下。

```
<?php
  echo "<pre>";
  echo "1) ".basename("/etc/sudoers.d",".d")."\n";
  echo "2) ".basename("/etc/sudoers.d")."\n";
  echo "3) ".basename("/etc/passwd")."\n";
```

```
  echo "4) ".basename("/etc/")."\n";
  echo "5) ".basename(".")."\n";
  echo "6) ".basename("/");
?>
```

运行程序，输出结果如下。

```
1) sudoers
2) sudoers.d
3) passwd
4) etc
5) .
6)
```

2. dirname()函数

dirname()函数用于返回路径中的目录部分。基本形式如下。

```
string dirname ( string $path )
```

给出一个包含有指向一个文件的全路径的字符串，本函数返回去掉文件名后的目录名。参数 $ path 是一个路径。在 Windows 中，斜线(/)和反斜线(\)都可以用作目录分隔符。在其他环境下是斜线(/)。

返回值：返回 $ path 的父目录。如果在 $ path 中没有斜线，则返回一个点('.')，表示当前目录。否则返回的是把 $ path 中结尾的 /component(最后一个斜线以及后面部分)去掉之后的字符串。

【示例 26】 使用 dirname()函数获得路径中的目录部分。

eg26.php 代码如下。

```
<?php
  echo "<pre>";
  echo "1) ".dirname("/etc/passwd")."\n";  // 1) /etc
  echo "2) ".dirname("/etc/")."\n";        // 2) / (or \ on Windows)
  echo "3) ".dirname(".");                 // 3) .
?>
```

运行结果如下。

```
1) /etc
2) \
3) .
```

3. pathinfo()函数

pathinfo()函数用于得到文件路径的信息。基本形式如下。

```
mixed pathinfo ( string $path [, int $options = PATHINFO_DIRNAME | PATHINFO_BASENAME |
PATHINFO_EXTENSION | PATHINFO_FILENAME ] )
```

pathinfo()返回一个关联数组包含有 $ path 的信息。返回关联数组还是字符串取决于 options。参数 $ path 是要解析的路径。参数 $ options 如果指定了，将会返回指定元素；它们包括 PATHINFO _ DIRNAME、PATHINFO _ BASENAME 和 PATHINFO _

EXTENSION 或 PATHINFO_FILENAME。如果没有指定 $ options，默认是返回全部的单元。

返回值：如果没有传入 $ options，将会返回包括以下单元的数组(array)：dirname、basename、extension(如果有)以及 $ filename。

【示例 27】 使用 pathinfo()函数获得文件路径的信息。

eg27. php 代码如下。

```
<?php
  echo "<pre>";
  $path_parts = pathinfo( '/www/htdocs/inc/lib.inc.php' );
  print_r($path_parts);
  //$path_parts 是数组，还可以写成如下形式：
  //echo $path_parts['dirname']."\n" ;
  //echo $path_parts['basename']."\n" ;
  //echo $path_parts['extension']."\n" ;
  //echo $path_parts['filename']          //从 PHP 5.2.0 开始
?>
```

程序运行结果如下。

```
Array
(
    [dirname] => /www/htdocs/inc
    [basename] => lib.inc.php
    [extension] => php
    [filename] => lib.inc
)
```

11.5.2　取得磁盘空间

在 PHP 中提供了获得磁盘空间的函数，主要是以下两个函数。

1. disk_total_space()函数

PHP 支持对磁盘容量的检查，包括 Win 32 系统和 UNIX 系统。PHP 提供的函数 disk_total_space()可以返回一个目录的磁盘总大小。基本形式如下。

```
float disk_total_space ( string $directory )
```

给出一个包含有一个目录的字符串，本函数将根据相应的文件系统或磁盘分区返回所有的字节数。本函数返回的是该目录所在的磁盘分区的总大小，因此在给出同一个磁盘分区的不同目录作为参数所得到的结果完全相同。在 UNIX 和 Windows 2000/XP/2008 中都支持将一个磁盘分区加载为一个子目录，这时正确使用本函数就很有意义。参数 $ directory 是文件系统的目录或者磁盘分区。

返回值：以浮点返回一个目录的磁盘总大小字节数，或者在失败时返回 false。

【示例 28】 使用 disk_total_space()函数获得磁盘空间。

eg28. php 代码如下。

```
<?php
```

```
  echo "<pre>";
  $ds1 =disk_total_space("/");
  //在 Windows 下
  $ds2 =disk_total_space("C:");
  $ds3 =disk_total_space("D:");
  echo $ds1."\n";
  echo $ds2."\n";
  echo $ds3;
?>
```

程序运行结果如下。

```
161067397120
64436809728
161067397120
```

2. disk_free_space()函数

disk_free_space()是 PHP 提供的另外一个获取磁盘空间信息的函数,它用于获取目录中的可用空间,其基本形式如下。

```
float disk_free_space ( string $directory )
```

disk_free_space()函数返回目录中的可用空间。给出一个包含有一个目录的字符串,本函数将根据相应的文件系统或磁盘分区返回可用的字节数。参数 $directory 表示文件系统目录或者磁盘分区。

返回值:以浮点返回可用的字节数,或者在失败时返回 false。

【示例 29】 使用 disk_free_space()函数获得磁盘剩余空间。

eg29.php 代码如下。

```
<?php
  echo "<pre>";
  //$df 包含根目录下可用的字节数
  $df =disk_free_space("/");
  //在 Windows 下
  echo $df_c =disk_free_space("C:"),"\n";
  echo $df_d =disk_free_space("D:");
?>
```

程序运行结果如下。

```
27698982912
72306847744
```

11.6 目录的基本操作

前面的章节中我们了解了使用有关函数获得目录的基本信息。本节将详细介绍目录的基本操作,包括目录的打开、关闭、读取、创建和解析。

11.6.1　目录的打开

在 PHP 中 opendir()函数用于打开目录的句柄，如果成功则返回目录句柄资源，失败则返回 false。基本形式如下。

```
resource opendir ( string $path [, resource $context ] )
```

打开一个目录句柄，可用于之后的 closedir()、readdir()和 rewinddir()的调用中。参数 $path 是要打开的目录路径，如果 path 不是一个合法的目录或者因为权限限制或文件系统错误而不能打开目录，opendir()返回 false 并产生一个 E_WARNING 级别的 PHP 错误信息。可以在 opendir()前面加上@符号来抑制错误信息的输出。

返回值：如果成功则返回目录句柄的 resource，失败则返回 false。

11.6.2　目录的关闭

目录打开之后若想再次关闭它，可以使用 closedir()函数。closedir()函数的基本形式如下。

```
void closedir ( resource $dir_handle )
```

该函数关闭由 $dir_handle 指定的目录流。流必须之前被 opendir()所打开。$dir_handle 参数是目录句柄。

返回值：无。

【示例 30】　使用 opendir()函数打开目录。

eg30.php 代码如下。

```
<?php
  echo "<pre>";
  $dir = "etc/php5/" ;
  if (is_dir($dir)){
    if ($dh = opendir($dir)){          //打开目录
      while (($file = readdir($dh)) !== false){
        echo "filename:  $file,  filetype:".filetype($dir.$file)."\n" ;
      }
      closedir ( $dh );                //关闭目录
    }
  }
?>
```

程序运行结果如下。

```
filename:  .,  filetype:dir
filename:  ..,  filetype:dir
```

说明：

- . 表示 etc，类型是目录。
- .. 表示 php5，类型也是目录。

11.6.3 目录的读取

opendir()函数用于打开磁盘上的一个目录,目录打开之后就可以使用 readdir()函数读取目录中的文件了,也可以进行复制、删除、修改操作。

1. readdir()函数

readdir()函数用于从目录句柄中读取条目。基本形式为:

```
string readdir ([ resource $dir_handle ] )
```

返回目录中下一个文件的文件名。文件名以在文件系统中的排序返回。参数 $dir_handle 是目录句柄的资源,之前由 opendir()打开。

返回值:成功则返回文件名,或者在失败时返回 false。

【示例 31】 使用 readdir()函数读取目录。

eg31.php 代码如下。

```
<?php
  echo "<pre>";
  if ($handle =opendir('path/to/files')){
    echo "Directory handle: $handle \n" ;
    echo "Files:\n" ;
    while (false  !== ($file =readdir($handle))){
      echo  " $file \n" ;
    }
    closedir ( $handle );
  }
?>
```

程序运行结果如下。

```
Directory handle: Resource id #3
Files:
  .
  ..
```

2. scanfdir()函数

在 PHP 中 scandir()函数列出指定路径中的文件和目录。基本形式为:

```
array scandir ( string $directory [, int $sorting_order [, resource $context ]] )
```

该函数返回一个 array,包含有 $directory 中的文件和目录。参数 $directory 是要被浏览的目录。$sorting_order 用来设置排序,默认的排序顺序是按字母升序排列。如果使用了可选参数 $sorting_order(设为 1),则排序顺序是按字母降序排列。$context 是可选参数,用于指定文件句柄环境。

返回值:成功则返回包含有文件名的数组,如果失败则返回 false。如果 $directory 不是目录,则返回布尔值 false 并生成一条 E_WARNING 级的错误。

【示例 32】 使用 scanfdir()函数读取目录中的子目录和文件。新建 eg32 文件夹,在该文件夹中新建文件夹 path 和文件 eg32.php,path 文件夹里面放置文件 11.gif 和 12.txt;在

path 文件夹下面新建文件夹 to，to 文件夹下面放置文件 php. gif，并新建文件夹 files；在 files 文件夹下面放置文件 php. gif 和 3. gif。

eg32/eg32. php 代码如下。

```
<?php
  echo "<pre>";
  $dir = 'path';                    //path 是存在的
  $files1 = scandir($dir);
  $files2 = scandir($dir,1);
  print_r($files1);
  print_r($files2);
?>
```

程序运行结果如下。

```
Array
(
    [0] => .
    [1] => ..
    [2] => 11.gif
    [3] => 12.txt
    [4] => to
)
Array
(
    [0] => to
    [1] => 12.txt
    [2] => 11.gif
    [3] => ..
    [4] => .
)
```

11.6.4　目录的创建

目录可以像文件一样创建，在 PHP 中可以使用函数 mkdir()来创建目录。基本形式如下。

```
bool mkdir ( string $pathname [, int $mode = 0777 [, bool $recursive = false
[, resource $context ]]] )
```

该函数尝试新建一个由 $pathname 指定的目录。参数 $pathname 是目录的路径；$mode 用于设置访问权，默认的 $mode 是 0777，意味着最大可能的访问权。$recursive 指定是否设置递归模式；$context 是文件的句柄环境。

【示例 33】　使用 mkdir()函数新建文件夹。新建文件夹 eg32，在该文件夹中新建网页 eg33. php。

eg33/eg33. php 代码如下。

```
<?php
  if (!mkdir("path",0,true)) {
```

```
        die('Failed to create folders path...' );
    }
    if (!mkdir("path/path1",0,true)){
        die('Failed to create folders path/path1...' );
    }
    if (!mkdir("path/path1/path2",0,true)) {
        die('Failed to create folders path/path1/path2...' );
    }
?>
```

说明：文件夹要一层一层地创建。

11.6.5 目录的删除

在 PHP 中空的目录可以使用函数 rmdir()来删除，但必须有删除权限。基本形式如下。

```
bool rmdir ( string $dirname [, resource $context ] )
```

在 PHP 中 rmdir()函数尝试删除 $ dirname 所指定的目录，该目录必须是空的，而且要有相应的操作权限。失败时会产生一个 E_WARNING 级别的错误。参数 $ dirname 是目录的路径。

返回值：删除成功时返回 true，或者在失败时返回 false。

【示例 34】 使用 rmdir()函数来删除目录。

eg34.php 代码如下。

```
<?php
    if (!is_dir('examples')){
        mkdir('examples');              //不存在则创建
    }
    rmdir('examples');                  //删除文件夹
?>
```

11.7 综合案例

本案例实现对文件夹和文件的基本操作。具体为创建文件夹，在文件夹下创建文件，并对文件进行读写操作。

步骤 1 在 ch11 文件夹中新建文件夹 eg35，并在该文件夹中新建页面 index.php，代码如下。

ch11/eg35/index.php 代码如下。

```
<?php
    header("content-type:text/html;charset=utf-8");
    echo "<pre>";
    $dir = "D:\\workdir";
    if (is_dir($dir)){
```

```
    echo "目录已经存在,无须再次创建。\n";
  }else{
    if (mkdir($dir))
       echo "创建目录{$dir}成功。";
    else
       echo "创建目录{$dir}失败。";
  }
?>
```

步骤 2　执行上面的代码,显示结果为:“创建目录 D:\workdir 成功”。

步骤 3　继续添加代码,用于在目录“D:\\workdir”中新建文本文件 test.txt,并在文件 test.txt 中添加文本。

ch11/eg35/index.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $dir = "D:\\workdir";
  if (is_dir($dir)){
    echo "目录已经存在,无须再次创建。\n";
  }else{
    if (mkdir($dir))
       echo "创建目录{$dir}成功。";
    else
       echo "创建目录{$dir}失败。";
  }
  //下面的代码是新添加的
  $filename = "D:\\workdir\\test.txt";
  $handle = fopen($filename,"w+");
  $text = <<<insert  //这是一些用于添加到 test.txt 中的文本,一共分为三行
  insert;
  $write = fwrite($handle,$text);
  fclose($handle);
?>
```

提示:$text 字符串中每一行都从最左边开始。若与上下行代码对齐,则被认为是空格字符。

执行上面的代码,自动创建文本文件 D:\\workdir\\test.txt,且在文本文件中写入内容,如图 11-5 所示。

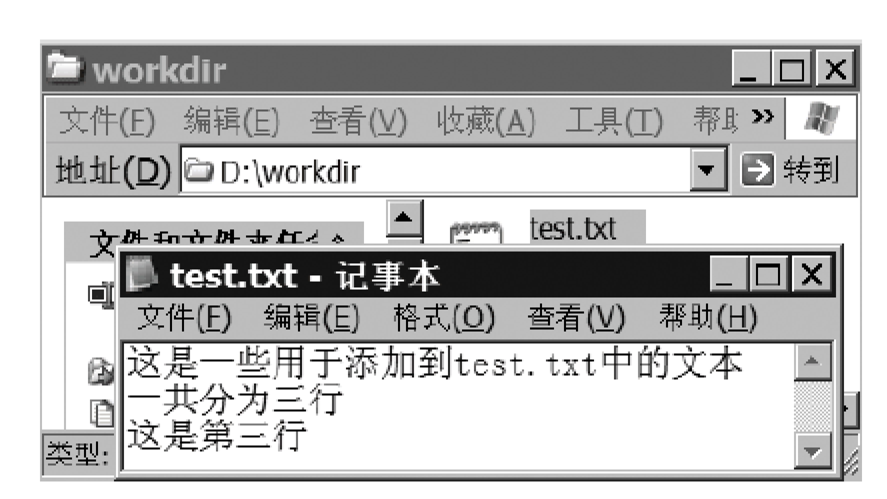


图 11-5　新建的文件 D:\\workdir\\test.txt

步骤 4 遍历目录 D:\\workdir 中的子目录和文件，并继续添加代码。

下面是 ch11/eg35/index.php 的完整代码：

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $dir ="D:\\workdir";
  if (is_dir($dir)){
    echo "目录已经存在，无须再次创建。\n";
  }else{
    if (mkdir($dir))
       echo "创建目录{$dir}成功。";
    else
       echo "创建目录{$dir}失败。";
  }
  $filename ="D:\\workdir\\test.txt";
  $handle =fopen($filename,"w+");
  $text =<<<insert
insert;
  $write =fwrite($handle,$text);
  fclose($handle);
  //下面的代码可遍历目录 D:\\workdir
  $handle =opendir($dir);
  while (false  !== ($file =readdir($handle))){
    echo  " $file \n" ;
  }
  closedir ( $handle );
?>
```

程序运行结果如下。

```
目录已经存在，无须再次创建。
  .
  ..
  test.txt
```

11.8 习 题

一、填空题

1. PHP 中提供的________函数用于获取文件的类型。

2. ________函数获取文件上次访问的时间，时间以 UNIX 时间戳形式返回；如果出错则返回 false。

3. 检查目录或者文件是否存在时，可以使用________函数。

4. ________函数从目录中读取条目，读取成功时返回目录下一个文件或者文件名，文件以文件系统中的排序返回。

5. ________函数可以将字符串写入文件。

二、选择题

1. 下面的说法中，________选项是错误的。
   A. filesize()函数用于获取文件的大小，执行成功返回 true，执行失败返回 false，并产生 E_WARNING 级别的错误
   B. PHP 中提供的函数 filesize()不仅可以获得文件的大小，还可以获取目录的大小
   C. PHP 中提供了与目录相关磁盘容量和可用空间的函数，但是没有提供与获取目录容量大小相关的函数
   D. 获取目录中的子目录之前，可以使用 is_dir()函数判断文件名是否为一个目录
2. 以只读方式打开，并且将文件指针指向文件头的代码是________。
   A. r　　B. r+　　C. w　　D. w+
3. 若服务器根目录下存在文件 test.txt，执行下面的代码，运行结果是________。

```
<?php
  $handle = fopen("test.txt","r");
  $data = fgets($handle,1);
  echo ftell($handle);
  fclose($handle);
?>
```

   A. 0　　B. 1　　C. 2　　D. 程序有误
4. PHP 中提供的________函数用于打开一个目录，________函数用于关闭目录。
   A. fopen(),fclose()　　B. openfile(),closefile()
   C. opendir(). closedir()　　D. dopen(),dclose()
5. 多次运行下面的代码，运行结果是________。

```
<?php
  $dirname = "works";
  $result =mkdir($dirname,10);
  if ($result)
    echo "OK";
  else
    echo "ERROR";
?>
```

   A. 输出 OK　　B. 输出 ERROR
   C. 第二行发生编译错误　　D. 输出 ERROR，并显示错误信息

三、程序设计题

编写程序来访问磁盘上某个目录，并遍历该目录，显示该目录中的子目录、文件的详细信息，如文件名、文件大小、文件类型、访问时间等。

第 12 章　MySQL 数据库

知识点：

- MySQL 数据库概述
- 数据库和数据表的创建
- 数据库服务器的连接
- 数据库的其他操作
- 数据库的数据操作
- 结构化查询语言
- 数据查询

本章导读：

网站一般要用到很多数据信息，需要使用数据库来保存。在 PHP 中使用 MySQL 数据库是最常见和方便的。

本章将讲述数据库操作、数据库数据操作以及结构化查询语句，最后还重点讲述了数据查询。

12.1　MySQL 数据库概述

数据库知识网站 DB-engines 更新了 2016 年 11 月的数据库流行度排行榜。毫无悬念，Oracle、MySQL 与 Microsoft SQL Server 依然霸占前三的位置。比起虽始终霸占榜首但积分大幅下降的 Oracle 来说，MySQL 在排行榜的积分不断增加，与 2015 年同期相比增加了 86.71 分，与 Oracle 的积分也只有不到 40 分的差距，大有赶超第一名的趋势。此外 MySQL 和 PHP 还是黄金搭档，因此本书介绍的数据库是基于 MySQL 数据库以及该数据库的操作类 mysqli。

12.1.1　MySQL 数据库的概念

MySQL 是一种开放源代码的关系型数据库管理系统（RDBMS），MySQL 数据库系统使用最常用的数据库管理语言——结构化查询语言（SQL）进行数据库管理。

由于 MySQL 是开放源代码的，因此任何人都可以在 General Public License 的许可下下载并根据个性化的需要对其进行修改。MySQL 因为其速度、可靠性和适应性而备受关注。大多数人都认为在不需要事务化处理的情况下，MySQL 是管理内容最好的选择。

MySQL 的海豚标志的名字叫 sakila，它是由 MySQL AB 的创始人从用户在“海豚命名”的竞赛中建议的大量的名字表中选出的。获胜的名字是由来自非洲斯威士兰的开源软

件开发者 Ambrose Twebaze 提供。根据 Ambrose 所说，Sakila 来自一种叫 SiSwati 的斯威士兰方言，也是在 Ambrose 的家乡乌干达附近的坦桑尼亚的 Arusha 的一个小镇的名字。

MySQL 虽然功能未必很强大，但因为它的开源、广泛传播，导致很多人都了解这个数据库，它的历史也富有传奇性。

MySQL 关系型数据库于 1998 年 1 月发行第一个版本。它使用系统核心提供的多线程机制提供完全的多线程运行模式，提供了面向 C、C++、Eiffel、Java、Perl、PHP、Python 以及 Tcl 等编程语言的编程接口(APIs)，支持多种字段类型并且提供了完整的操作符支持查询中的 SELECT 和 WHERE 操作。

MySQL 是开放源代码的，因此任何人都可以在 General Public License 的许可下下载并根据个性化的需要对其进行修改。MySQL 因为其速度、可靠性和适应性而备受关注。

12.1.2　MySQL 服务器的启动、连接、断开和停止

MySQL 数据库服务器的主要操作有启动 MySQL 服务器、连接和断开 MySQL 服务器以及停止 MySQL 服务器。

1. 启动 MySQL 服务器

在 Windows 操作系统中启动 MySQL 服务器的方式主要有两种：通过 phpStudy 2016 对话框启动 MySQL 服务器，或者在命令提示符下输入命令启动它。

(1) 命令提示符下输入命令启动 MySQL 服务器

选择“开始”→“运行”命令，在弹出的“运行”窗口中输入 cmd 命令，按 Enter 键进入命令提示符窗口。在命令提示符下输入“D:”，进入 D 盘；然后输入 cd phpStudy\MySQL\bin，进入 MySQL 所在文件夹；再输入 mysqld install 安装相关服务；输入 net start mysql，此时服务启动，如图 12-1 所示。

提示：只有启动了 MySQL 服务，才能输入类似 mysql-root-root 命令进入 MySQL 命令提示符。MySQL 服务器路径与读者计算机的安装方式有关系。



图 12-1　启动 MySQL 服务

(2) 通过 phpStudy2016 对话框启动 MySQL 服务器

双击桌面上的 phpStudy2016 图标或选择“开始”→“程序”→phpStudy 命令，出现如图 12-2 所示对话框。右击“启动”按钮，选择“启动 MySQL”命令，即可启动 MySQL 服务，

或者单击“启动”按钮，可以同时启动 Apache 服务器(网站服务器)和 MySQL 服务器。启动后的服务呈现绿色圆点状态。

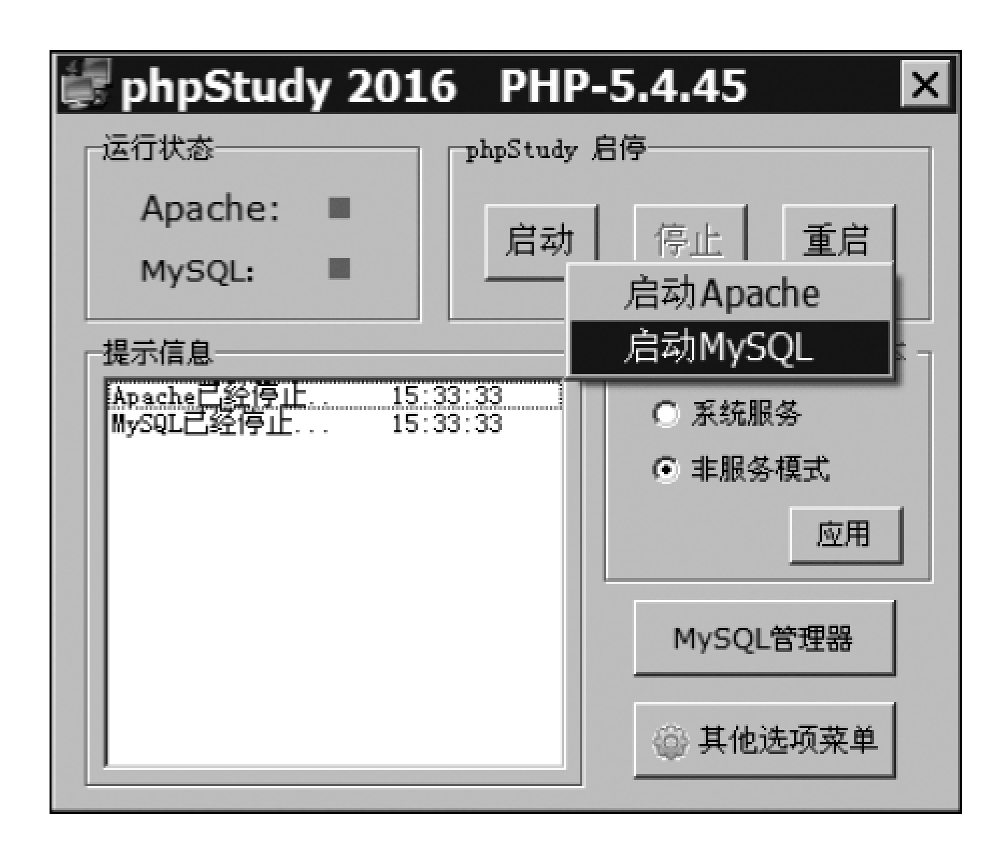


图 12-2 使用该对话框启动 MySQL 服务

2. 连接和断开 MySQL 服务器

(1) 连接 MySQL 服务器

启动 MySQL 服务之后才能连接 MySQL 服务器。进入 mysql 所在目录，在命令提示符下输入 mysql-h 主机名-u 用户名-p 密码”。此时进入 MySQL 命令符(mysql>)，表示连接 MySQL 服务器成功，如图 12-3 所示。

提示：此处用户名和密码都是 root，用户根据自己的实际情况输入用户名和密码，连接本机服务器可以省略 h 项，直接输入“mysql-u 用户名-p 密码”，例如可以输入 mysql-uroot -proot。h、p 和 u 字符之后不要加空格。

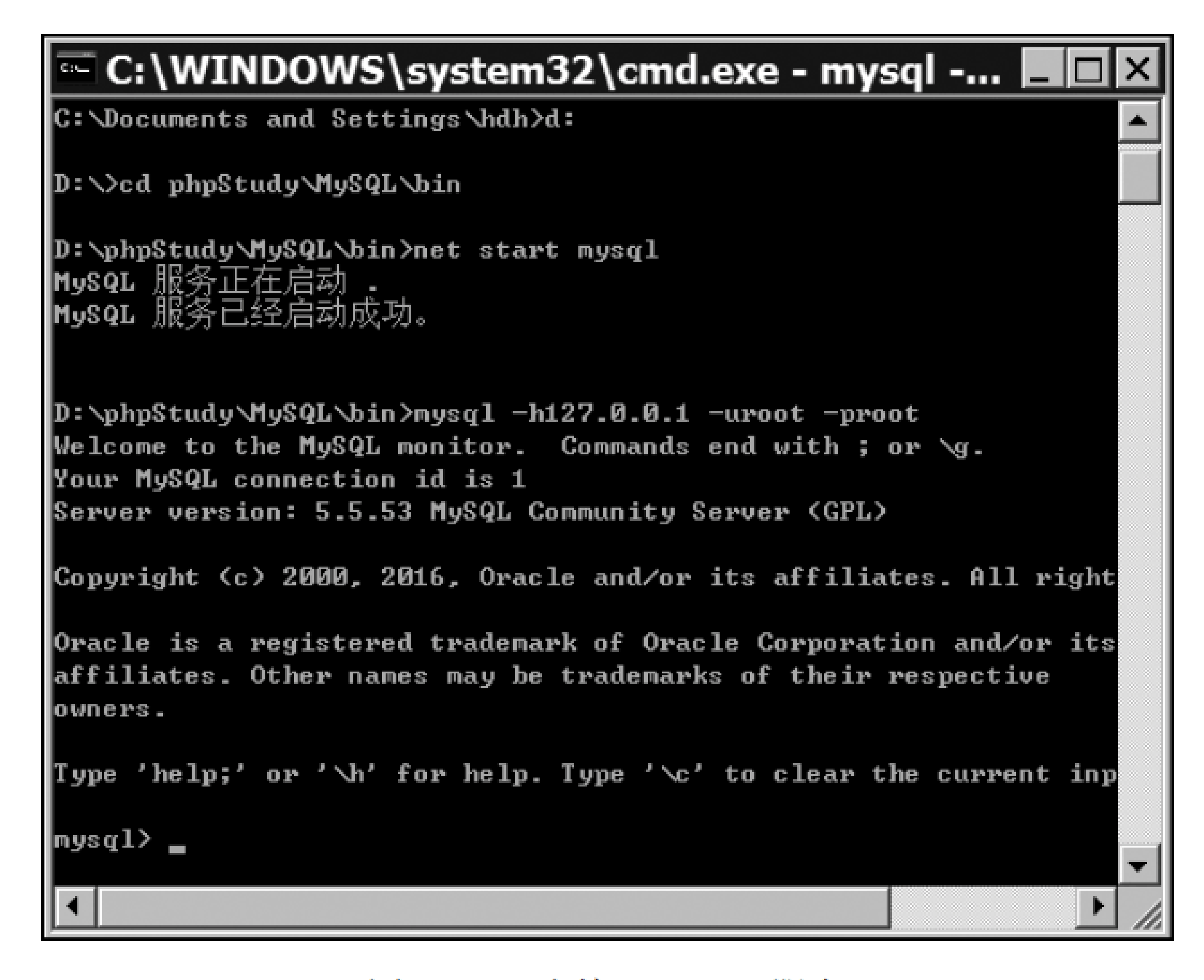


图 12-3 连接 MySQL 服务

(2) 断开 MySQL 服务器

连接 MySQL 服务后，可以通过在 MySQL 提示符下输入 exit 命令断开 MySQL 连接，

如图 12-4 所示。

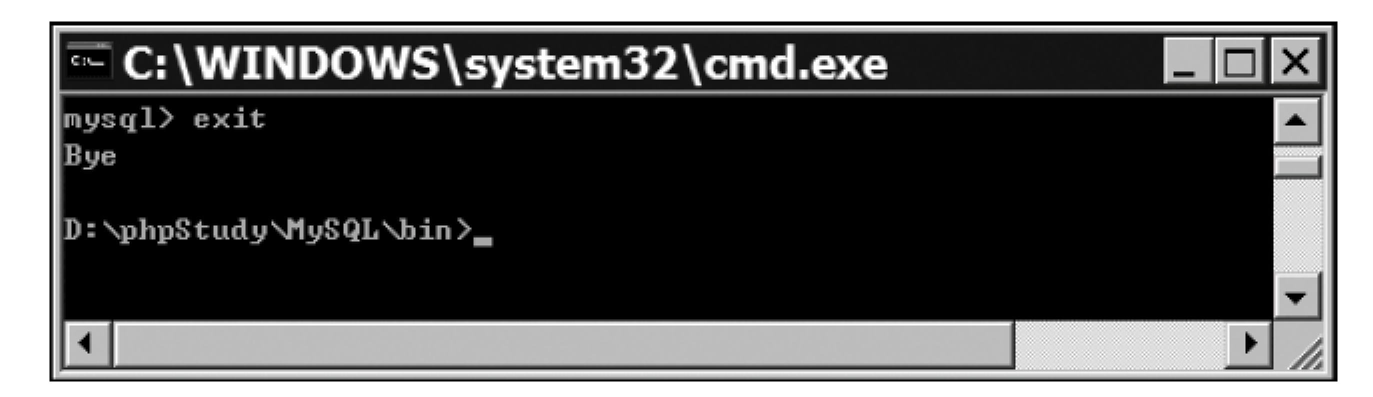


图 12-4　断开 MySQL 服务

3. 停止 MySQL 服务器

(1) 通过 phpStudy 2016 PHP-5.4.45 对话框停止 MySQL 服务器

在 phpStudy 2016 PHP-5.4.45 对话框中，右击“停止”按钮，选择“停止 MySQL”命令，即可停止 MySQL 服务器。还可以单击“停止”按钮同时停止 Apache 服务器和 MySQL 服务器。根据所需来选择，如图 12-5 所示。停止后的 MySQL 服务器图标呈现红色正方形状态。

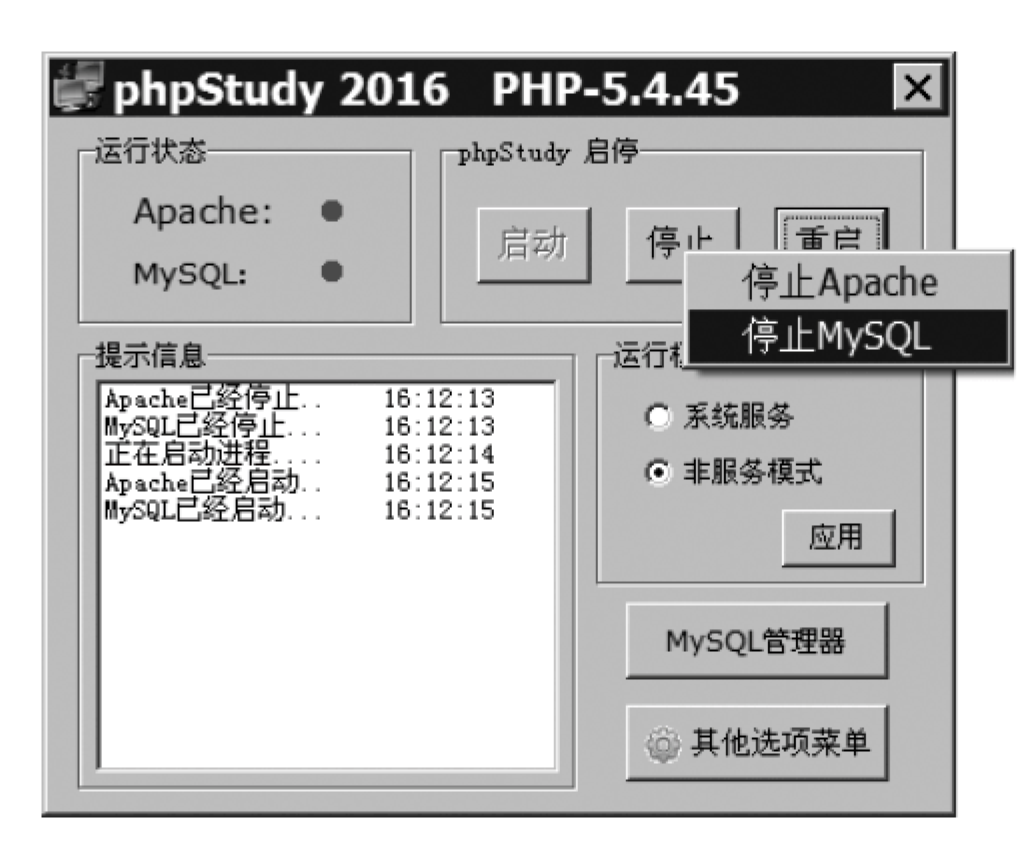


图 12-5　停止 MySQL 服务器

(2) 在命令行输入代码来停止 MySQL 服务器

在 Windows 命令行下进入 D:\phpStudy\MySQL\bin，再输入 net stop mysql 命令，即可停止 MySQL 服务，如图 12-6 所示。

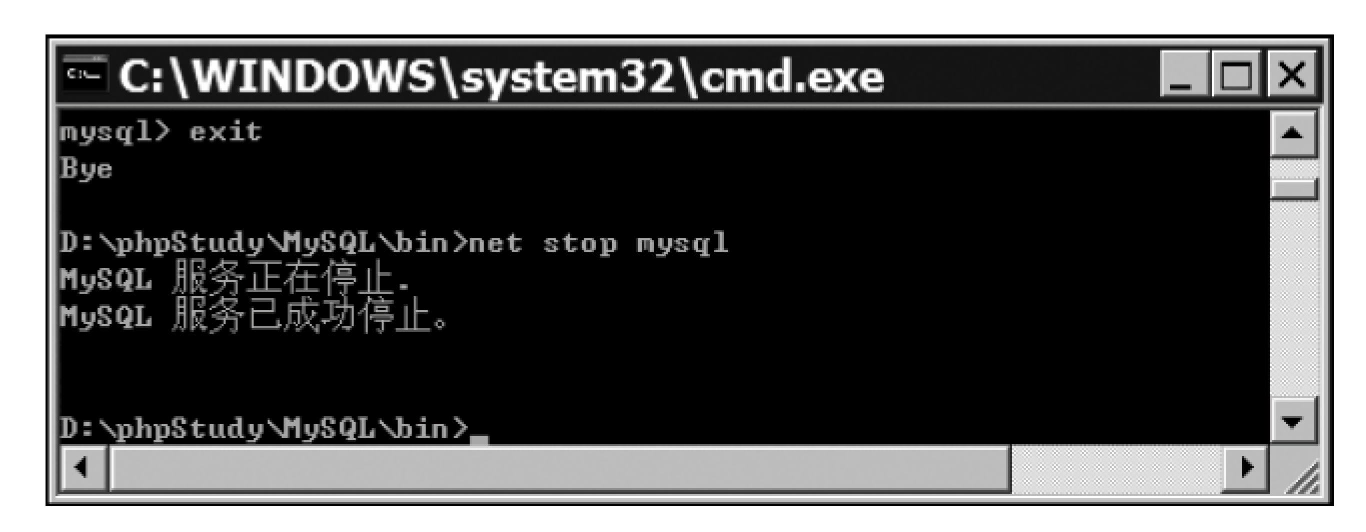


图 12-6　停止 MySQL 服务

12.1.3 数据库常用类

在 PHP 中可以使用 MySQL 类来处理数据库连接以及其他数据库操作，但是由于使用 MySQL 连接要防止 SQL 注入问题，而且相对来说比较慢，于是就有了 mysqli 这个基于 MySQL 的优化类。

mysqli 类的优化效果非常明显。使用预处理方式完全解决了 SQL 注入问题，但是它的缺点是只支持 PHP 5.0 以上的版本。

PDO 是最新出来的一种，连接方式兼容大部分数据库，也解决了 SQL 注入问题，但是也只支持 PHP 5.0 以上的版本。但是 PDO 执行效率不如 mysqli 类。因此本书将讲述 mysqli 类，并以此来进行数据库、数据表、记录等各种操作。表 12-1 给出了 mysqli 类的方法。

表 12-1　mysqli 类的方法

方法名称	说　明
__construct()	构造方法，用于创建一个新的 mysqli 对象，也可以建立一个连接
autocommit()	开启或关闭数据库修改并自动提交
change_user	改变了数据库连接所指定的用户
character_set_name()	返回数据库连接默认的字符集
close()	关闭先前打开的连接
commit()	提交当前的事务
connect()	打开一个到 MySQL 数据库服务器的新的连接
debug()	执行调试操作
dump_debug_info()	转储调试信息
get_client_info()	返回客户端版本
get_host_info()	返回一个字符串代表的连接使用类型，如 Localhost via UNIX socket
get_server_info()	返回 MySQL 服务器的版本，如 4.1.2-alpha-debug
get_server_version()	返回整数形式的 MySQL 服务器版本，如 40102
init()	初始化 mysqli 并返回一个资源
info()	检索有关最近执行的查询
kill()	要求服务器去杀死一个 MySQL 线程
multi_query()	执行多个查询语句
more_results()	从多查询语句中检查是否有更多的查询结果
next_result()	从当前执行的多查询中读取下一个结果
options()	设置选项
ping()	如果没有连接，ping 一台服务器连接或重新连接
prepare()	准备一个 SQL 语句的执行，返回 mysqli_stmt 对象
query()	与数据库的任何交互都是通过查询进行的，该方法向数据库发送查询来执行
real_connect()	试图打开一个到 MySQL 数据库服务器的连接
escape_string()	转义特殊字符的字符串，用于一个 SQL 语句，并考虑到当前的字符集的连接
rollback()	回滚当前的事务

续表

方法名称	说明
select_db()	为数据库查询选择默认的数据库
set_charset()	设置默认客户端字符集
ssl_set()	使用 SSL 建立安全连接
stat()	获取当前的系统状态
stmt_init()	初始化一个声明,并返回一个 mysqli_stmt 对象
store_result()	从最后查询中转让结果集
thread_safe()	可考虑返回安全的线程

表 12-1 中列出的方法既有用于数据库连接的方法,又有对数据库、对表、对记录进行其他操作的方法。此外 mysqli 类还提供了一些属性,用于对 MySQL 数据库进行各种操作。表 12-2 列出了 13 个 mysqli 类的属性。

表 12-2 mysqli 类的属性

属性名称	说明
$ affected_rows	在前一个 MySQL 操作中获取影响的行数
$ client_info	MySQL 客户端版本返回为一个字符串
$ client_version	MySQL 客户端版本返回为一个整数
$ errno	返回最近函数调用的错误代码
$ error	返回最近函数调用的错误信息字符串
$ field_count	返回最近查询获取的列数
$ host_info	返回一个字符串的连接类型
$ info	检索最近执行的有关查询
$ insert_id	返回使用最后查询自动生成的编号
$ protocol_version	返回 MySQL 协议使用的版本
$ sqlstate	返回一个字符串,包含 SQLSTATE 错误码的最后一个错
$ thread_id	为当前连接返回线程 ID
$ warning_count	返回前一个 SQL 语句执行过程中产生的警告数量

12.2 数据库以及数据表的创建

数据库(Database)是按照数据结构来组织、存储和管理数据的仓库,它产生于六七十年前,随着信息技术和市场的发展,特别是 20 世纪 90 年代以后,数据管理不再仅仅是存储和管理数据,而转变成用户所需要的各种数据管理的方式。数据库有很多种类型,从最简单的存储有各种数据的表格到能够进行海量数据存储的大型数据库系统,都在各个方面得到了广泛的应用。

本节将使用命令法和 phpMyAdmin 界面创建数据库和数据表,以及添加数据表记录。如果读者安装的是其他软件,也可以使用相应软件创建 MySQL 数据库。

至于数据库、数据表、记录、表结构的各种操作，将在后面章节中使用代码来实现。本节不做赘述。

12.2.1 使用命令创建 MySQL 数据库

本书作者安装的是 phpStudy 2016 多合一软件，其中集成了 PHP、MySQL、Apache、phpMyAdmin 等软件。创建 MySQL 数据库的命令语法为：

```
create database <database_name>;
```

MySQL 命令中每一个语句都要加一个分号。如果不加分号，写成多行也是可以的。本语句负责建立一个名为 database_name 的 MySQL 数据库，习惯上 MySQL 数据库都以 db_ 为前缀。要想创建 MySQL 数据库，首先要进入 MySQL 命令行。

【示例 1】 在命令行中创建数据库 db_db1。

步骤 1 首先要进入 MySQL 命令行状态。

mysql.exe 文件是可执行的文件，负责创建 MySQL 数据库等操作，该命令可以在 DOS 环境下运行，因此要先进入该文件所在目录。该文件通常在目录 D:\phpStudy\MySQL\bin 中。在 Windows“运行”对话框中输入 cmd，单击“确定”按钮，或者使用“Windows 键(在键盘上有个 Windows 标志的按键)+R”组合键输入 cmd 后按 Enter 键。如图 12-7 所示为“运行”对话框。



图 12-7 Windows 命令行输入 cmd

步骤 2 切换目录至 D:\phpStudy\MySQL\bin。

此时，进入 DOS 操作界面，切换到 D 盘，如图 12-8 所示。

然后在 D:\>盘符后输入命令 cd phpStudy\MySQL\bin，此时出现的界面如图 12-9 所示，此时已经进入目录 D:\phpStudy\MySQL\bin。



图 12-8 切换至 D 盘根目录



图 12-9 进入目录 D:\phpStudy\MySQL\bin

步骤 3 输入 mysql.exe 命令进入 MySQL 操作界面。

输入命令：mysql-uroot -proot。注意大小写不要写错，不要人为在-u 和-p 之后加空格。这里的-u 后面紧跟的是用户名 root，-p 后面紧跟的是密码 root。对于安装路径和密码不相同的用户而言，上述操作略有不同(路径和密码的不同)。

此时命令提示符显示为 mysql>，表示已经进入 MySQL 操作界面。

步骤 4 创建数据库 db_db1。

输入命令“create database db_db1;”。注意 MySQL 中的命令都是要在最后加分号的，如图 12-10 所示。

步骤 5　显示创建好的数据库。

在 MySQL 命令行中输入命令“show databases;”，该命令仅仅是为了显示出全部数据库，这样可以确认刚才创建的数据库 db_db1 是存在的，如图 12-11 所示。



图 12-10　数据库 db_db1 已经创建成功



图 12-11　显示创建好的数据库

12.2.2　使用 phpMyAdmin 界面创建 MySQL 数据库

phpMyAdmin 界面可以用于创建数据库、创建表、对数据库进行各种操作、对表进行各种操作，操作非常方便，无须死记硬背 MySQL 命令。

【示例 2】　使用 phpMyAdmin 界面创建 MySQL 数据库 db_db2。

步骤 1　启动浏览器。在网址中输入 http://localhost/phpMyAdmin/或者输入 http://127.0.0.1/phpMyAdmin/，出现如图 12-12 所示的界面，此界面用于输入用户名和密码。我们输入用户名 root 以及密码 root。



图 12-12　登录 phpMyAdmin 界面

之所以这样操作，是因为在目录 D:\phpStudy\WWW 下面存在子目录 phpMyAdmin，且该目录中存在页面 index.php。

步骤 2 输入用户名和密码后出现 phpMyAdmin 界面。在该界面中选择“数据库”，并在“新建数据库”文本框中输入数据库名 db_db2，单击“创建”按钮，此时数据库 db_db2 将被新建(会出现提示信息“创建数据库 db_db2 成功”)，如图 12-13 所示。



图 12-13 创建数据库 db_db2

12.2.3 使用命令创建 MySQL 数据表

在 MySQL 中要想创建数据表，首先要创建数据库并选择数据库。创建数据表使用 create table 命令。MySQL 中 create table 语句的基本语法是：

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    [(create_definition,...)][column_definition]
    [table_options] [select_statement]
```

- TEMPORARY：该关键字表示用 MySQL create table 新建的表为临时表，此表在当前会话结束后将自动消失。临时表主要应用于存储过程中，对于目前尚不支持存储过程的 MySQL，该关键字一般不用。
- IF NOT EXISTS：实际上是在建表前加上一个判断，只有该表目前尚不存在时才执行 create table 操作。用此选项可以避免出现表已经存在而无法再新建的错误。
- tbl_name：要创建的表的名称。该表名必须符合标识符规则，通常的做法是在表名中仅使用字母、数字及下画线，例如 titles、our_sales、my_user1 等都应该算是比较规范的表名。
- create_definition：这是 create table 语句中的关键部分所在。在该部分具体定义了表中各列的属性。
- column_definition：对列的属性的定义。格式如下。

```
col_name type [NOT NULL | NULL] [DEFAULT default_value]
    [AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]
```

```
[COMMENT 'string']
```

✍col_name：表中列的名字。必须符合标识符规则，而且在表中要唯一。

✍type：列的数据类型。有的数据类型需要指明长度，并用括号括起。

✍NOT NULL | NULL：指定该列是否允许为空。如果既不指定 NULL 也不指定 NOT NULL，列被认为指定了 NULL。

✍DEFAULT default_value：为列指定默认值。如果没有为列指定默认值，MySQL 会自动分配一个。如果列可以取 NULL 作为值，则默认值是 NULL。如果列被声明为 NOT NULL，默认值取决于列类型：

  ✍对于没有声明 AUTO_INCREMENT 属性的数字类型，默认值是 0；对于一个 AUTO_INCREMENT 列，默认值是在顺序中的下一个值。

  ✍对于除 TIMESTAMP 的日期和时间类型，默认值是该类型适当的“零”值；对于表中第一个 TIMESTAMP 列，默认值是当前的日期和时间。

  ✍对于除 ENUM 的字符串类型，默认值是空字符串；对于 ENUM，默认值是第一个枚举值。

✍AUTO_INCREMENT：设置该列有自增属性，只有整型列才能设置此属性。当插入 NULL 值或 0 到一个 AUTO_INCREMENT 列中时，列被设置为 value+1，在这里 value 是此前表中该列的最大值。AUTO_INCREMENT 顺序从 1 开始。每个表只能有一个 AUTO_INCREMENT 列，并且它必须被索引。

✍UNIQUE：在 UNIQUE 索引中，所有的值必须互不相同。如果在添加新行时使用的关键字与原有行的关键字相同，则会出现错误。

✍KEY：KEY 通常是 INDEX 的同义词。如果关键字属性 PRIMARY KEY 在列定义中已给定，则 PRIMARY KEY 也可以只指定为 KEY。这么做的目的是与其他数据库系统兼容。

✍PRIMARY KEY：表示一个唯一 KEY，此时所有的关键字列必须定义为 NOT NULL。如果这些列没有被明确地定义为 NOT NULL，MySQL 应隐含地定义这些列。一个表只有一个 PRIMARY KEY。如果没有 PRIMARY KEY 并且一个应用程序要求在表中使用 PRIMARY KEY，则 MySQL 返回第一个 UNIQUE 索引，此索引没有作为 PRIMARY KEY 的 NULL 列。

✍COMMENT：对于列的说明可以使用 COMMENT 选项来进行指定。说明通过 SHOW CREATE TABLE 和 SHOW FULL COLUMNS 语句显示。

- table_option：格式如下。

```
{ENGINE|TYPE} = engine_name
| AUTO_INCREMENT = value
| AVG_ROW_LENGTH = value
| [DEFAULT] CHARACTER SET charset_name [COLLATE collation_name]
| CHECKSUM = {0 | 1}
| COMMENT = 'string'
| CONNECTION = 'connect_string'
| MAX_ROWS = value
| MIN_ROWS = value
```

```
| PACK_KEYS = {0 | 1 | DEFAULT}
| PASSWORD = 'string'
| DELAY_KEY_WRITE = {0 | 1}
| ROW_FORMAT = {DEFAULT|DYNAMIC|FIXED|COMPRESSED|REDUNDANT|COMPACT}
| UNION = (tbl_name[,tbl_name]...)
| INSERT_METHOD = { NO | FIRST | LAST }
| DATA DIRECTORY = 'absolute path to directory'
| INDEX DIRECTORY = 'absolute path to directory'
```

ENGINE 和 TYPE 选项用于为表指定存储引擎。ENGINE 是首选的选项名称。ENGINE 和 TYPE 选项采用值见表 12-3。

表 12-3　ENGINE 和 TYPE 选项及说明

存储引擎	说　明
ARCHIVE	档案存储引擎
BDB	带页面锁定的事务安全表,也称为 BerkeleyDB
CSV	值之间用逗号隔开
EXAMPLE	示例引擎
FEDERATED	可以访问远程表的存储引擎
(OBSOLETE)ISAM	在 MySQL 5.1 中没有此引擎。如果要从以前的版本升级到 MySQL 5.1,应该在进行升级前把原有的 ISAM 表转换为 MyISAM 表
InnoDB	带行锁定和外键的事务安全表
MEMORY	本表类型的数据只保存在存储器里(在早期 MySQL 版本中被称为 HEAP)
MERGE	MyISAM 表的集合,作为一个表使用。也称为 MRG_MyISAM
MyISAM	二进制轻便式存储引擎,此引擎是 MySQL 所用的默认存储引擎
NDBCLUSTER	成簇表,容错表,以存储器为基础的表。也称为 NDB

【示例 3】 使用命令法在数据库 db_db1 中创建表 tb_tbl。

步骤 1　进入 MySQL 命令行,进入 D:\phpStudy\MySQL\bin 目录,并输入 mysql-uroot -proot。具体操作步骤见示例 1。

步骤 2　在 MySQL 命令行中输入命令"show databases;",可以在列表中看到数据库 db_db1 的存在。

步骤 3　设置 db_db1 数据库为默认数据库。在 MySQL 命令行中输入"use db_db1;"并按 Enter 键。显示信息 Database changed,表明已经改变了当前的数据库,如图 12-14 所示。再输入命令 show tables,显示该数据库中有哪些表。显示信息 Empty set (0.05sec),表明该数据库中还没有表存在,如图 12-15 所示。



图 12-14　改变数据库

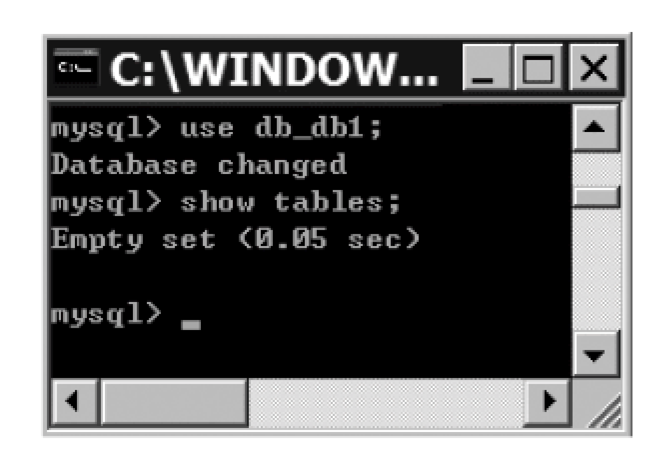


图 12-15　查看数据库中的表

步骤 4　输入命令建表。

在 MySQL 命令行中输入命令：

```
CREATE TABLE  `tb_tbl` (
`id` INT( 4 ) NOT NULL AUTO_INCREMENT ,
  `user` VARCHAR( 20 ) NOT NULL ,
  `password` VARCHAR( 10 ) NOT NULL ,
  `yuwen` INT( 3 ) NOT NULL ,
  `sex` VARCHAR( 6 ) NOT NULL DEFAULT  'male',
  `addr` VARCHAR( 90 ) NOT NULL ,
PRIMARY KEY (  `id` )
) ENGINE =MYISAM DEFAULT CHARSET =utf8;
```

图 12-16 所示为命令的输入结果。

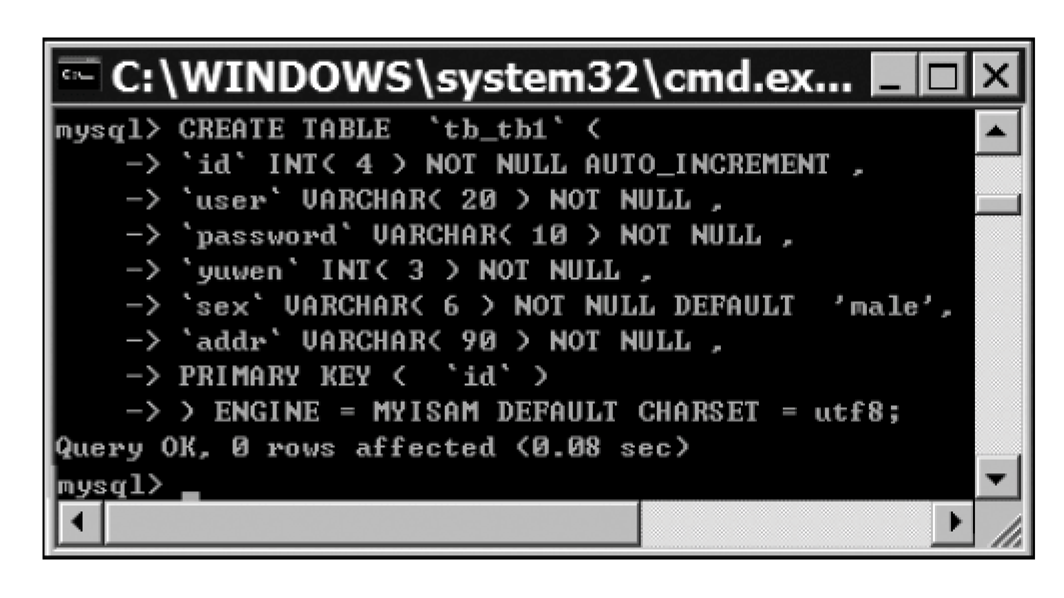


图 12-16　在数据库 db_db1 中创建表 tb_tbl

说明：

- “CREATE TABLE `tb_tbl` (”表明要建立的表名是 tb_tbl。
- “`id` INT(4) NOT NULL AUTO_INCREMENT ,”表明建立字段是 id，后面的参数是字段数据类型；AUTO_INCREMENT 表示字段自动加 1。
- “`user` VARCHAR(20) NOT NULL ,”表明要建立字段 user，字符串类型，长度小于或等于 20，非空。
- “`password` VARCHAR(10) NOT NULL ,”表明要建立字段 password，字符串类型，长度小于或等于 10，非空。
- “`yuwen` INT(3) NOT NULL ,”表明要建立字段 yuwen，整型类型，长度为 3，非空。
- “`sex` VARCHAR(6) NOT NULL DEFAULT 'male',”表明建立字段 sex，字符串类型，长度为小于或等于 6，非空，默认值为 male。
- “`addr` VARCHAR(90) NOT NULL ,”表明要建立字段 addr，类型为字符串，长度小于或等于 90，非空。
- “PRIMARY KEY (`id`)”设置 id 字段为主键，不允许任何两条记录的 id 相同，也不任何记录的 id 为空。
- “) ENGINE = MYISAM DEFAULT CHARSET = utf8;”设置存储引擎和编码。

步骤 5　查看表。

输入命令“show tables;”，显示如图 12-17 所示信息，说明



图 12-17　建好的表 tb_tbl

表 tb_tb1 已经建好。

本步骤并非必需，仅仅是查看数据表是不是已经建好，当然还可以使用命令 “show full fields from tb_tb1;”查看表结构，使用命令“select * from tb_tb1;”查看表记录。

至此，数据表已经建好。

12.2.4 使用 phpMyAdmin 界面创建 MySQL 数据表

要想使用 phpMyAdmin 界面创建 MySQL 数据表，必须使用用户名和密码登录到 phpMyAdmin 界面。在创建并选择某个数据库后，再在该数据库中创建数据表。

【示例 4】 使用 phpMyAdmin 界面在数据库 db_db2 中创建数据表 tb_tb2。

步骤 1 参考示例 2 打开 phpMyAdmin 界面，然后在左边导航栏中选中数据库 db_db2，如图 12-18 所示。



图 12-18 选择数据库 db_db2

步骤 2 在打开界面右边工作区的“新建数据表”中的“名字”文本框中输入要新建的数据表表名 tb_tb1，在“字段数”文本框中输入字段个数为 6。然后单击“执行”按钮，进入如图 12-19 所示界面。



图 12-19 建表界面

步骤 3 在界面中输入各个字段的结构参数，如图 12-20 所示，然后单击“保存”按钮。此时表 tb_tb1 已经建立完毕。

步骤 4 至此，完成表 tb_tb1 的新建。根据需要还可以在此界面中修改表结构。首先

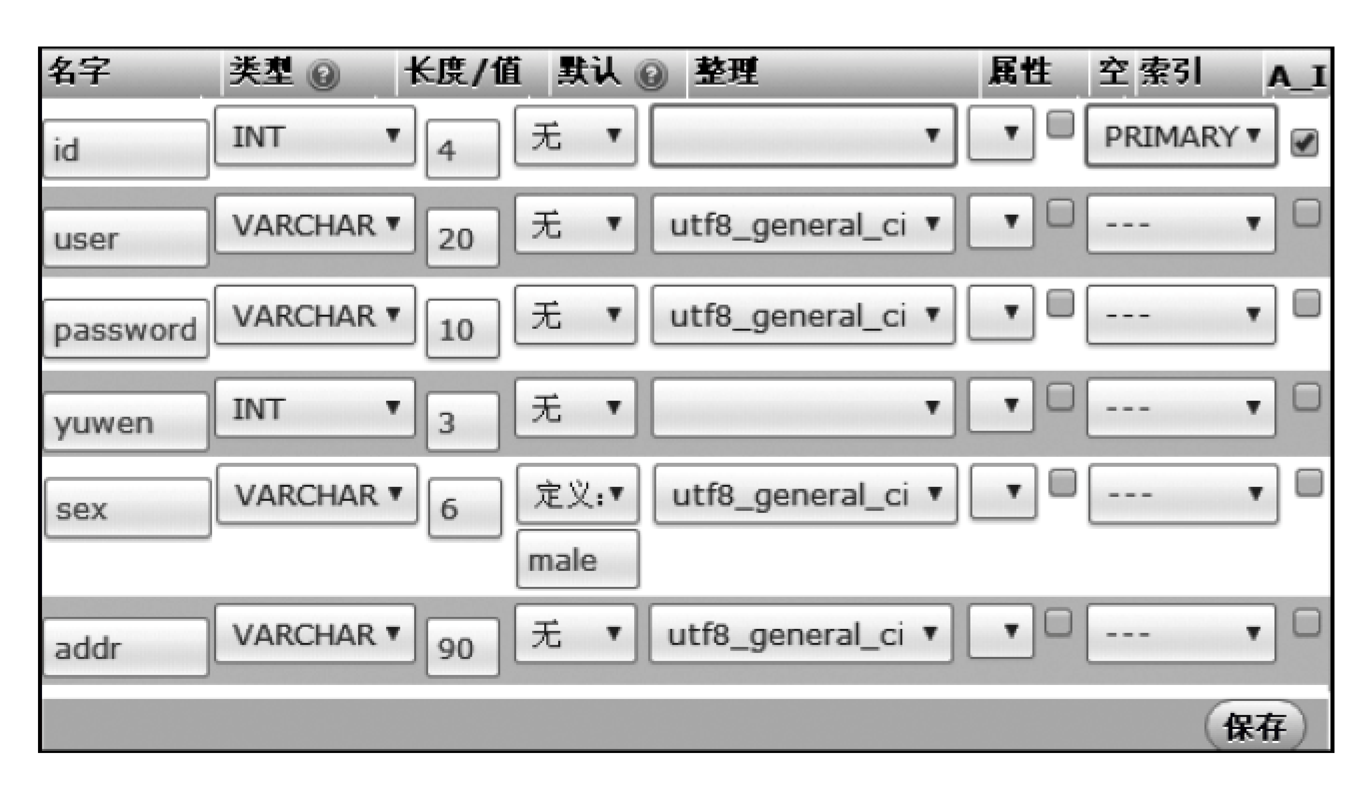


图 12-20　新建表 tb_tbl

选中数据库，再选中表，然后在右边工作区中选择“结构”，选择想要修改的字段后，可以单击“修改”图标进行表结构的修改，如图 12-21 所示。



图 12-21　tb_tbl 表的结构

12.2.5　使用命令在表中添加记录

在 MySQL 中可以使用 insert 语句进行记录的添加。该语句的语法格式为：

```
INSERT INTO table_name (列 1, 列 2,...) VALUES (值 1, 值 2,...)
```

向表 table_name 中添加记录，值 1、值 2 分别对应于列 1、列 2。在添加记录时需要满足主键非空且唯一，以及满足值列表和对应的字段列表数据类型的匹配性。

【示例 5】　使用代码在数据库 db_db1 的表 tb_tbl 中添加一条记录或多条记录。

步骤 1　进入 MySQL 命令界面，输入如下代码：

```
USE db_db1;
INSERT INTO `db_db1`.`tb_tbl`
  (`id`,`user`,`password`,`yuwen`,`sex`,`addr`)
VALUES
(NULL,'liubei','123456','90','male','wuhan');
```

如图 12-22 所示为输入命令的效果。

```
C:\WINDOWS\system32\cmd.exe - mysql ...
mysql> USE db_db1;
Database changed
mysql> INSERT INTO  `db_db1`.`tb_tb1`
    -> (`id` ,`user` ,`password` ,`yuwen` ,`sex` ,`addr`)
    -> VALUES
    -> (NULL ,  'liubei',  '123456',  '90',  'male',  'wuhan');
Query OK, 1 row affected (0.00 sec)
```

图 12-22　插入一条记录

步骤 2　输入如下代码来查看插入的记录：

```
select * from tb_tbl;
```

效果如图 12-23 所示。

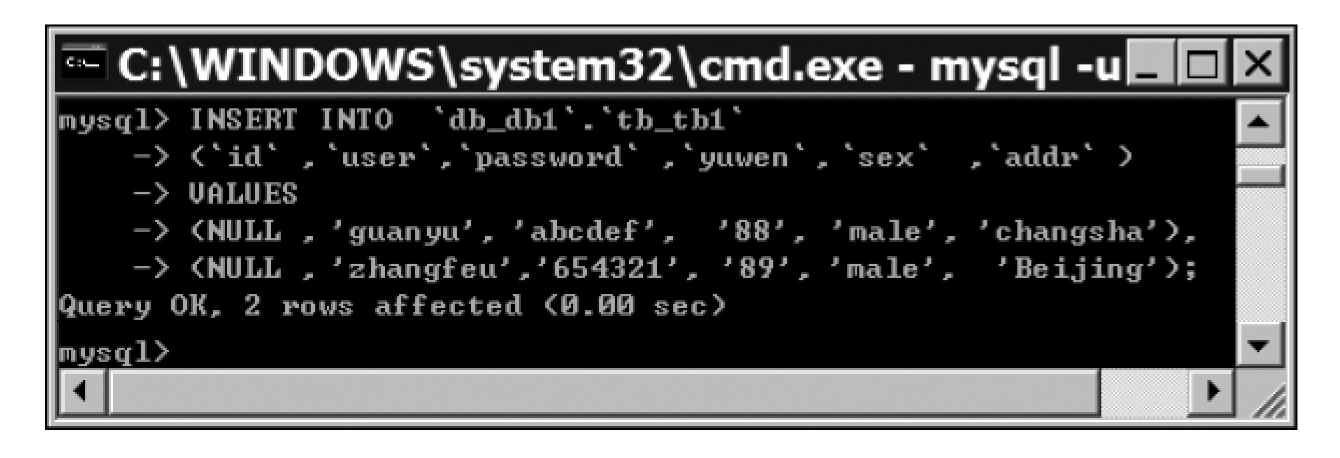
```
C:\WINDOWS\system32\cmd.exe - mysql ...
mysql> select * from tb_tb1;
+----+--------+----------+-------+------+-------+
| id | user   | password | yuwen | sex  | addr  |
+----+--------+----------+-------+------+-------+
|  1 | liubei | 123456   |    90 | male | wuhan |
+----+--------+----------+-------+------+-------+
1 row in set (0.00 sec)
```

图 12-23　查看插入的记录

步骤 3　还可以一次性插入多条记录，输入如下代码：

```
INSERT INTO  `db_dbl`.`tb_tbl`
(`id` ,  `user` ,`password` ,`yuwen` ,`sex` ,`addr` )
VALUES
(NULL ,  'guanyu',  'abcdef',  '88',  'male',  'changsha'),
(NULL ,  'zhangfeu',  '654321',  '89',  'male',  'Beijing');
```

效果如图 12-24 所示。



```
C:\WINDOWS\system32\cmd.exe - mysql -u
mysql> INSERT INTO  `db_db1`.`tb_tb1`
    -> (`id` ,`user`,`password` ,`yuwen`,`sex`  ,`addr` )
    -> VALUES
    -> (NULL ,'guanyu','abcdef',  '88', 'male', 'changsha'),
    -> (NULL ,'zhangfeu','654321', '89', 'male',  'Beijing');
Query OK, 2 rows affected (0.00 sec)
mysql>
```

图 12-24　一次性插入多条记录

查看记录，如图 12-25 所示。

说明：

- 因为 id 是主键，且能够自动加 1，所以在插入记录时，不考虑 id；id 将会自动从 1 开始，每次自动加 1，取值会是 1、2、3…。
- 因为 sex 有默认值，在插入记录时也可以不考虑，系统会自动使用默认值。
- VALUES 值列表和字段列表的个数和顺序必须一一对应，数据类型且要匹配。

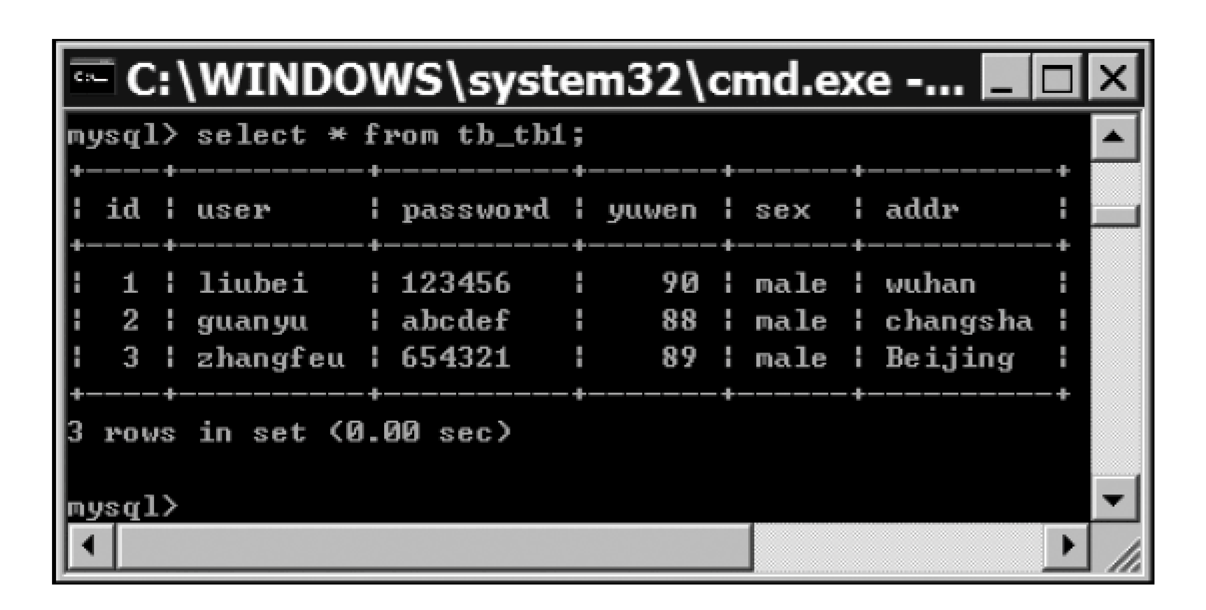


图 12-25 查看插入的记录

因此，下面的写法都是正确的：

```
INSERT INTO  `db_dbl`.`tb_tbl`
( `user` ,`password` ,`yuwen` ,`sex` ,`addr` )       //无 id
VALUES ( 'guanyu', 'abcdef', '88', 'male', 'changsha');
```

和

```
INSERT INTO `db_dbl`.`tb_tbl`
( `user` ,`password` ,`yuwen` ,`addr` )              //无 id 和 sex
VALUES ( 'guanyu', 'abcdef', '88', 'changsha');
```

12.2.6 使用 phpMyAdmin 界面在数据表中添加记录

phpMyAdmin 界面可以用于创建数据库、创建表、对数据库进行各种操作、对表进行各种操作，包括表记录的添加，操作非常方便。

【示例 6】 使用 phpMyAdmin 界面在数据库 db_db2 的表 tb_tbl 中添加多条记录。

步骤 1 参考示例 2 进入 phpMyAdmin 界面。

步骤 2 在如图 12-26 所示的界面的左边数据库列表中选中数据库 db_db2，然后再选中表 tb_tbl。



图 12-26 插入表记录

步骤 3 在右边工作区中选择“插入”菜单，出现如图 12-27 所示的插入记录界面。输入多条记录，然后单击“执行”按钮即可完成记录的添加。

步骤 4 插入记录并单击“执行”按钮后，可以单击“浏览”按钮查看插入的记录，如图 12-28 所示。

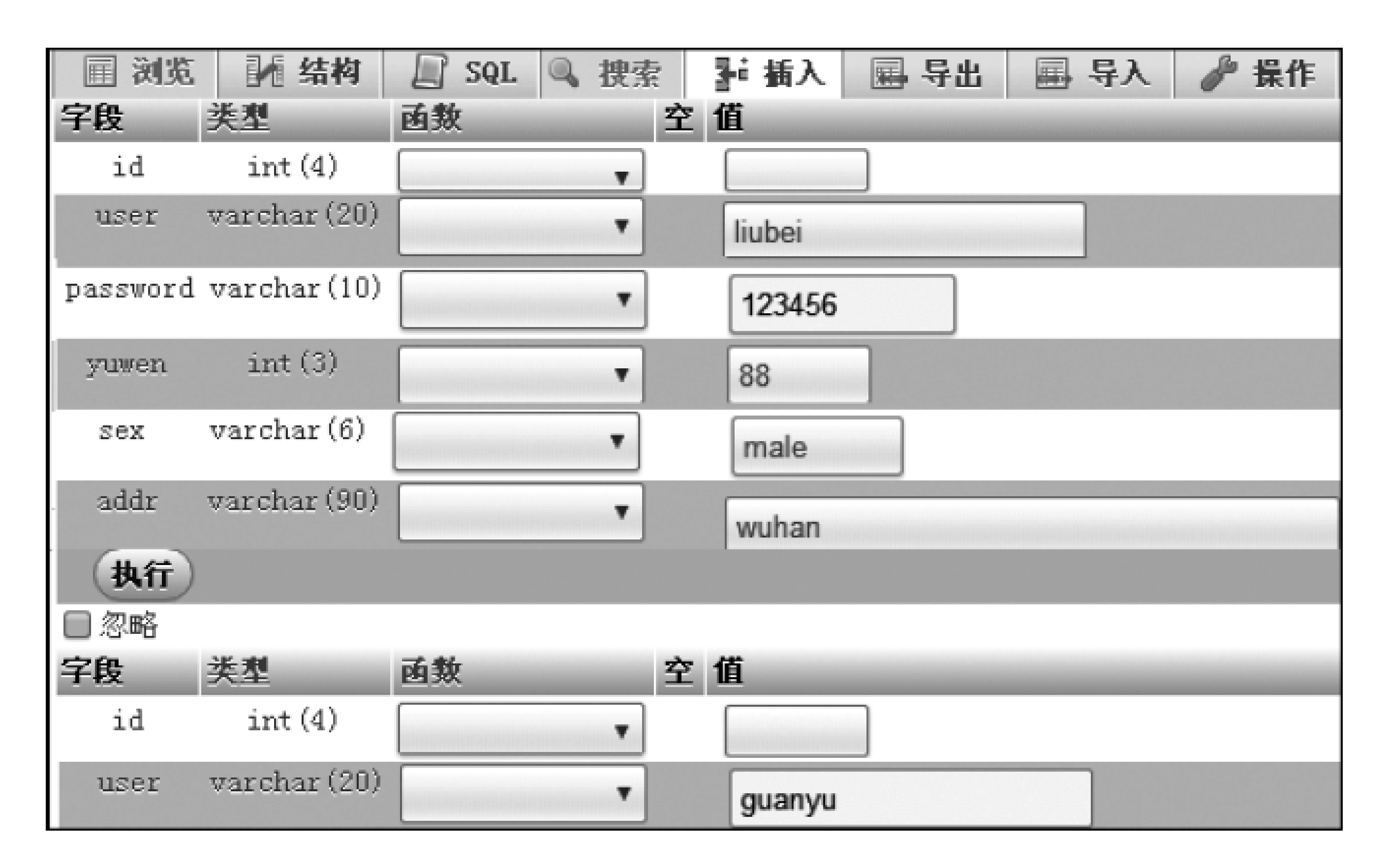


图 12-27　插入记录界面



图 12-28　查看插入的记录

12.3　数据库服务器的连接

可以使用 mysqli 类的构造方法来创建连接对象，还可以使用 mysqli_init()方法来创建连接，使用 mysqli_connect_errno()方法来验证连接错误。

12.3.1　连接对象的创建

创建数据库连接对象有多种方法。这里介绍 3 种方法。第一种是直接创建没有参数的连接对象，再使用其他方法指定连接数据库、服务器、用户名和密码。第二种是指定服务器、用户名和密码，再用其他方法指定数据库名。第三种方法是指定数据库服务器、用户名、密码、数据库名。

【示例 7】　连接本地数据库服务器，用户名和密码都是 root，数据库名是 db_db1，端口是 3306。有如下 3 种方法。

方法 1：

eg7-1.php 代码如下。

```
<?php
  $mysqli = new MySQLi();                                        //创建连接对象
  $mysqli->connect('127.0.0.1', 'root', 'root', 'db_db1',3306);  //连接服务器,选择数据库
  //或$mysqli->connect('127.0.0.1', 'root', 'root', 'db_db1');
?>
```

方法 2：

eg7-2.php 代码如下。

```
<?php
  $mysqli = new MySQLi('127.0.0.1', 'root', 'root');   //创建连接对象,顺便连接服务器
  $mysqli->select_db('db_db1');                        //选择数据库
?>
```

方法 3：

eg7-3 代码如下。

```
<?php
  $mysqli = new MySQLi();                              //创建连接对象
  $mysqli->connect('127.0.0.1', 'root', 'root');       //连接服务器
  $mysqli->select_db('db_db1');                        //选择数据库
?>
```

在实际应用中常常还需要设置字符集。

eg7.php 代码如下。

```
<?php
  $mysqli = new MySQLi('127.0.0.1', 'root', 'root', 'db_db1', '3306');
  $mysqli->set_charset("utf8");
?>
```

12.3.2　设置连接选项

使用构造方法创建连接对象时无法设置 MySQL 特有的连接选项。如果想要设置连接选项，可以使用 mysqli_init()方法来创建连接，然后再使用 options()方法来设置数据库选项，最后再使用 real_connect()方法与指定的数据库建立连接。

mysqli_init()方法用于创建连接对象。该函数的语法格式为：

```
mysqli_init(void)
```

返回值：返回连接对象。

options()方法用于设置连接对象的连接选项，该函数的语法格式如下。

```
mysqli_options(connection,option,value);
```

该函数用于设置连接选项。

有关参数说明如下。

(1) connection：必需，规定要使用的 MySQL 连接。

(2) option：必需，规定要设置的选项。可以是下列值中的一个。

- MYSQLI_OPT_CONNECT_TIMEOUT：以秒为单位的连接超时时间。
- MYSQLI_OPT_LOCAL_INFILE：启用/禁用 LOAD LOCAL INFILE。
- MYSQLI_INIT_COMMAND：在连接到 MySQL 服务器之后执行的命令。
- MYSQLI_READ_DEFAULT_FILE：从已命名的文件而不是 my.cnf 中读取选项。
- MYSQLI_READ_DEFAULT_GROUP：从 my.cnf 或者 MYSQLI_READ_DEFAULT_FILE 中指定的文件中的已命名组中读取选项。
- MYSQLI_SERVER_PUBLIC_KEY：基于 SHA-256 认证的 RSA 公共密钥文件。

(3) value：必需，规定 option 的值。

返回值：如果成功则返回 true，如果失败则返回 false。

【示例 8】 连接数据库服务器，并设置连接选项。

eg8.php 代码如下。

```
<?php
  $mysqli =mysqli_init();
  $mysqli->options(MYSQLI_OPT_CONNECT_TIMEOUT , 2);
  //设置超时时间
  $mysqli->options(MYSQLI_INIT_COMMAND , 'SET COMMANDMIT=0');
  //连接成功即执行'SET COMMANDMIT=0'
  $mysqli->real_connect('127.0.0.1' , 'root' , 'root ' , 'db_db1');
?>
```

12.3.3 连接错误测试

在数据库程序执行其他操作之前最好先测试一下连接数据是否有错，比如要连接的服务器不存在或者数据库不存在，则程序无须向下继续运行。

(1) mysqli_connect_errno()方法用于测试连接。该函数(或属性)语法格式如下。

面向对象风格：

```
string $mysqli->connect_errno ;
```

过程化风格：

```
int mysqli_connect_errno ( void );
```

该函数测试连接错误，无须参数。

返回值：无错误则返回 0，有错误则返回错误序号。

(2) mysqli_connect_error()方法可以得到连接的错误信息。该函数的语法格式为：

面向对象风格：

```
string $mysqli->connect_error ;
```

过程化风格：

```
string mysqli_connect_error ( void )
```

返回值：返回描述错误的信息，没有错误则返回 NULL。

【示例 9】 测试连接错误信息。eg9-1.php 采用过程化风格，eg9-2.php 采用面向对象的风格。还可以使用其他属性来获得错误信息，如 eg9-3.php。

eg9-1.php 代码如下。

```
<?php
  $mysqli =mysqli_init();
  $mysqli->options(MYSQLI_OPT_CONNECT_TIMEOUT, 2);
  //设置超时时间
  $mysqli->options(MYSQLI_INIT_COMMAND,'SET AUTOCOMMIT=0');
  //连接成功即执行'SET COMMANDMIT=0'
  $mysqli->real_connect('127.0.0.1', 'root', 'root', 'db_dblx');  //不存在数据库 db_dblx
  if (mysqli_connect_errno()){
    die ( ' 连接失败: ' . mysqli_connect_error());       //显示错误信息,不再向下执行
  }else{
    echo '连接成功!';
  }
?>
```

eg9-2.php 代码如下。

```
<?php
  $mysqli =mysqli_init();
  $mysqli->options( MYSQLI_OPT_CONNECT_TIMEOUT , 2);     //设置超时时间
  $mysqli->options(MYSQLI_INIT_COMMAND , 'SET AUTOCOMMIT=0');
  //连接成功即执行'SET COMMANDMIT=0'
  $mysqli->real_connect('127.0.0.1', 'root', 'root', 'db_dblx');
  if  ( $mysqli->connect_errno ) {
    die (' 连接失败 : ' . $mysqli->connect_error);
  }else{
    echo '连接成功!';
  }
?>
```

eg9-3.php 代码如下。

```
<?php
  $mysqli =mysqli_init();
  $mysqli->options(MYSQLI_OPT_CONNECT_TIMEOUT, 2);      //设置超时时间
  $mysqli->options(MYSQLI_INIT_COMMAND,'SET AUTOCOMMIT=0');
  //连接成功即执行'SET COMMANDMIT=0'
  $mysqli->real_connect('127.0.0.1', 'root', 'root', 'db_dblx');
  if ($mysqli->errno){                                   //返回最近函数调用的错误代码
    die ('连接失败:'.$mysqli->error);                    //返回最近函数的错误信息
  }else{
    echo '连接成功!';
  }
?>
```

以上 3 个程序运行结果均为：

```
Warning: mysqli::real_connect(): (HY000/1049): Unknown database 'db_dblx' in D:\phpStudy\WWW
\ch13\eg9-2.php on line 5
连接失败:Unknown database 'db_dblx'
```

12.3.4　连接的关闭

虽然面向对象的程序设计中,一个对象在不使用后能够自动析构并释放相应的连接对象,但是在较长的程序代码中,不使用的连接对象最好使用代码人工关闭并释放该连接对象,因为 PHP 系统的析构只有在程序执行的最后时刻进行。不使用的连接对象越早释放越好,使用 mysqli 对象提供的 close()函数可以关闭数据库连接,成功则返回 true,否则会返回 false。

【示例 10】　连接数据库服务器并在获取数据库服务器相关信息之后关闭连接。

eg10.php 代码如下。

```
<?php
  echo "<pre>";
  $mysqli = new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");       //设置字符集
  var_dump($mysqli->get_charset());  //显示字符集信息
  $mysqli->close();                   //关闭连接
  var_dump($mysqli->get_charset() ); //关闭连接之后再也无法正常显示字符集信息
  //产生警告信息: mysqli::get_charset(): Couldn't fetch mysqli in
  //D:\phpStudy\WWW\ch13\eg10.php on line 7
  //得到 NULL
?>
```

12.4　数据库的其他操作

启动 MySQL 服务器后,即可以对 MySQL 数据库进行操作。数据库的操作主要包括创建数据库、查看数据库、选择数据库、删除数据库。新建数据库在前面章节已经阐述。

12.4.1　查看数据库

MySQL 中可以使用 SHOW DATABASES 来查看数据库,如图 12-29 所示。

还可以在 PHP 中编写代码查看数据库。

【示例 11】　查看数据库。

eg11.php 代码如下。

```
<?php
  echo '<pre>';
  $mysqli = new MySQLi('127.0.0.1','root','root');
  $result = $mysqli->query('show databases');
  while($row = $result->fetch_row()){
    $data[] = $row[0];
  }
```

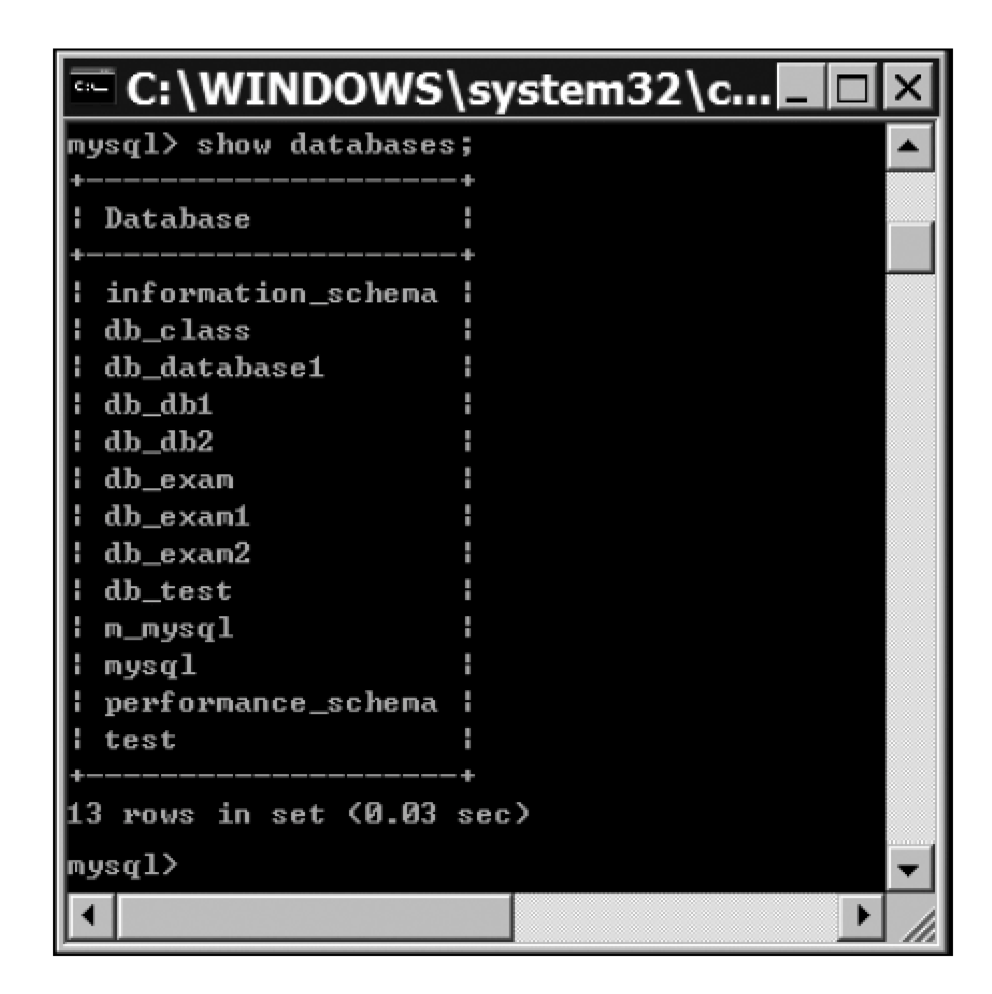


图 12-29　查看数据库

```
  var_dump($data);
?>
```

提示：有关 mysqli 类和 mysqli_result 类可以参见后面章节。

程序运行结果如图 12-30 所示。



图 12-30　查看数据库

12.4.2 选择数据库

在创建数据库之后，还可以在数据库中新建数据表或进行其他操作，但是在创建表之间最好要先选择数据库，也就是让某个数据库成为当前默认数据库。MySQL 中使用“USE 数据库名”进行数据库的选择。例如，选择数据库 db_database1 可以使用命令 use db_database1，如图 12-31 所示。

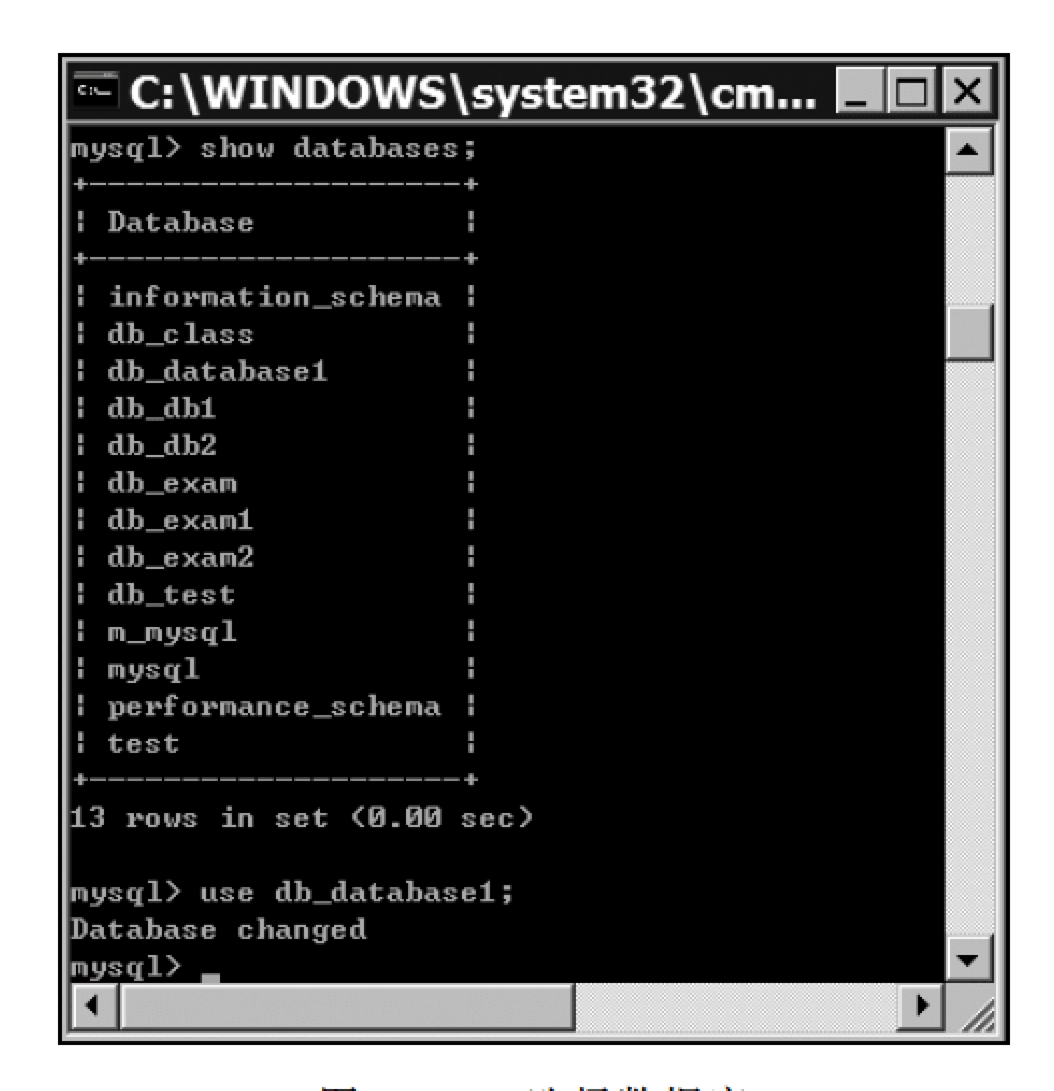


图 12-31 选择数据库

12.4.3 删除数据库

不用的数据库可以删除，在 MySQL 中可以使用命令“DROP DATABASE 数据库名”进行数据库的删除。

提示： 删除数据库应该谨慎，一旦删除数据库，数据库中的结构、表等各种信息将全部被删除，不可恢复。

12.5 数据库数据的操作

mysqli 类能够操作数据库，比如执行查询获得表记录。但是涉及数据库中具体的数据时则需要使用 mysqli_result 类。例如，使用 mysqli 类查询表记录后可以返回一个 mysqli_result 类，然后需要得到查询到的每个记录时则可以使用 mysqli_result 类提供的方法。

12.5.1 mysqli 类

在 mysqli 类中提供了几种执行 SQL 命令的方法，其中最常见的是 query()方法。使用 query()方法可以对数据库表进行增、删、改、查等操作。mysqli 类提供了较多的数据库操作方法和属性，详见表 13-1 和表 13-2。

mysqli 类提供的 affected_rows 属性可以获取有多少个记录受到影响。

【示例 12】　向数据库 db_db1 的表 tb_tbl 中添加两条记录。

eg12. php 代码如下。

```
<?php
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr ="insert into tb_tbl(`user`,`password`,`yuwen`,`sex`,`addr`)
  values ('Caocao','123456','99','male','Xuchang'),('Sunquan','654321','87',
  'male','Nanjing')";
  if ( $mysqli->query($sqlstr) )
     echo $mysqli->affected_rows.'2条记录受到影响!';
  $mysqli->close();
?>
```

上述代码执行结果如下。

```
2条记录受到影响!
```

【示例 13】　查询数据库 db_db1 的表 tb_tbl 中语文大于 80 分的记录。

eg13. php 代码如下。

```
<?php
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr ="select * from tb_tbl where `yuwen`>80";
  if ( $mysqli->query($sqlstr) )
     echo $mysqli->affected_rows.'2条记录受到影响!';
  $mysqli->close();
?>
```

本程序虽然可以查询语文分数大于 80 的记录，且能得到大于 80 的记录数，但是却无法得到这些查询到的记录，这需要配合 mysqli_result 类。

12.5.2　mysqli_result 类

在 mysqli 类执行 query()等方法时，如果要对返回值进行操作，则需要使用 mysqli_result 类。该类提供了多个方法和属性用于对返回数据进行操作。换言之，mysqli_result 类能够处理多个有返回值的 SQL 语句，并对 SQL 语句所返回结果集进行处理。mysqli_result 类提供的方法和属性分别见表 12-4 和表 12-5。

表 12-4　mysqli_result 类提供的方法

方 法 名 称	说　　明
close()	释放内存并关闭结果集
data_seek()	明确改变当前结果记录顺序
fetch_field()	从结果集中获取某一个字段的信息
fetch_fields()	从结果集中获取所有字段的信息
fetch_field_direct()	从一个指定的列中获取列详细信息，返回一个包含列信息的对象

续表

方 法 名 称	说　　明
fetch_array()	将以一个普通索引数组和关联数组两种形式返回一条结果记录
fetch_assoc()	将以一个普通的关联数组的形式返回一条结果记录
fetch_object()	将以一个对象的形式返回一条结果记录
fetch_row()	将以一个普通的索引数组的形式返回一条结果记录
field_seek()	设置结果集中字段的偏移位置

表 12-5　mysqli_result 类提供的属性

属 性 名 称	说　　明
$ current_field	获取当前结果中指向的字段偏移位置，是一个整数
$ field_count	从查询结果中获取列的个数
$ lengths	返回一个数组，保存在结果集中获取当前行的每一个列的长度
$ num_rows	返回结果集中包含记录的行数

mysqli_result 类的对象默认是通过 mysqli 对象并在使用 query()方法执行 select 查询语句时返回的，并把结果保存在该 mysqli_result 对象中。

12.5.3　获取数据记录的方法

mysqli_result 类的 fetch_array()、fetch_assoc()、fetch_object()和 fetch_row()四个方法用于从结果集中依次读取数据记录。它们仅仅在字段的引用方面有细微的差别。它们的共同点是每次都读取并返回当前的记录(记录指针所在的记录)，然后将记录指针指向下一行记录。读取记录过程中，如果记录指针已经到达最后一条记录的后面则无法读取记录，此时返回 false。

1. fetch_row()方法

从结果集中读取并返回当前记录，然后将记录指针指向下一行记录。读取记录过程中，如果记录指针已经到达最后一条记录的后面则无法读取记录，此时返回 false。因此可以使用循环读取全部记录。fetch_row()方法获得的记录以索引数组的形式存在，所以还可以与 list()函数结合起来使用。

【示例 14】 从数据库 db_db1 的表 tb_tbl 中读取记录。

eg14. php 代码如下。

```
<?php
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr ="select * from tb_tbl";
  $mysqli_result =$mysqli->query($sqlstr);
  //执行 query()方法的 select 语句并返回 mysqli_result 类的对象
  $row =$mysqli_result->fetch_row();          //读取第一条记录
  for ($i=0 ; $i<count($row) ; $i++)          //输出每一个字段
    echo $row[$i]." ";
  echo "<br />";
  $row =$mysqli_result->fetch_row();          //读取第二条记录
```

```
  for ( $i=0 ; $i<count($row) ; $i++)          //输出每一个字段
    echo $row[$i]." ";
  echo "<br />";
  list($id,$xuehao,$xingming) =$mysqli_result->fetch_row();
  //读取第三条记录,将数组前三个元素赋值给 list()函数的变量
  echo $id." ".$xuehao." ".$xingming."<br />";
  $mysqli_result->data_seek(0);
  //记录指针置为第一条记录,这是必需的,否则会从第四条记录开始读
  while($row =$mysqli_result->fetch_row()){ //通过循环读取全部记录
    for ( $i=0 ; $i<count($row) ; $i++)        //输出每一个字段
      echo $row[$i]." ";
    echo "<br />";
  }
  $mysqli_result->close();
  $mysqli->close();
?>
```

上述代码执行结果如下。

```
1 liubei 123456 90 male wuhan
2 guanyu abcdef 88 male changsha
3 zhangfei 654321
1 liubei 123456 90 male wuhan
2 guanyu abcdef 88 male changsha
3 zhangfei 654321 88 male Beijing
```

2. fetch_assoc()方法

从结果集中读取并返回当前记录,然后将记录指针指向下一条记录。读取记录过程中,如果记录指针已经到达最后一条记录的后面则无法读取记录,此时返回 false。因此可以使用循环读取全部记录。fetch_assoc()方法获得的记录以关联数组的形式存在,所以它无法与 list()函数结合起来使用。

【示例 15】 从数据库 db_db1 的表 tb_tbl 中读取记录。

eg15.php 代码如下。

```
<?php
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr ="select * from tb_tbl";
  $mysqli_result =$mysqli->query($sqlstr);
  //执行 query()方法的 select 语句并返回 mysqli_result 类的对象
  $row =$mysqli_result->fetch_assoc();          //读取第一条记录
  echo $row['id']." ".$row['user']." ".$row['password']
  ." ".$row['yuwen']." ".$row['sex']." ".$row['addr']."<br />";
  $row =$mysqli_result->fetch_assoc();          //读取第二条记录
  echo $row['id']." ".$row['user']." ".$row['password']
  ." ".$row['yuwen']." ".$row['sex']." ".$row['addr']."<br />";
  $mysqli_result->data_seek(0);
  //记录指针置为第一条记录,否则会从第四条记录开始读
  while($row =$mysqli_result->fetch_assoc()){
  //通过循环读取全部记录,并保存到二维数字$data,$data[0]是第一条记录
```

```
    $data[] = $row;
  }
  //循环输出该二维数组。count($data)为记录数
  for($i=0;$i<count($data);$i++){
  //还可以把 count($data)写成$mysqli_result->num_rows。num_rows 表示结果集记录数
    echo $data[$i]['id'].' '. $data[$i]['user'].' '.$data[$i]['password'].'
     ';
    echo $data[$i]['yuwen'].' '. $data[$i]['sex'].' '.$data[$i]['addr']."<br />";
  }
  $mysqli_result->close();
  $mysqli->close();
?>
```

上述代码执行结果如下。

```
1 liubei 123456 90 male wuhan
2 guanyu abcdef 88 male changsha
1 liubei 123456 90 male wuhan
2 guanyu abcdef 88 male changsha
3 zhangfei 654321 88 male Beijing
```

3. fetch_array()方法

fetch_array()方法是 fetch_row()和 fetch_assoc()两种方法的结合版,所得结果集为关联数组或者数字索引数组,或者可以同时为关联数组和索引数组。

默认情况下,fetch_array()方法会同时获得两种数组。当然用户还可以通过设置参数来获得不同类型的数组。fetch_array()方法的语法格式为:

```
array fetch_array([array_type])
```

该方法与前面讲述的 fetch_row()和 fetch_assoc()两种方法类似,不同之处在该函数可以带参数。

有关参数说明如下。

- MYSQL_ASSOC:关联数组;
- MYSQL_NUM:数字数组;
- MYSQL_BOTH:默认值,同时产生关联和数字数组。

返回值:返回关联数组,或数字数组,或关联数组和数值数组。

【示例 16】 从数据库 db_db1 的表 tb_tbl 中读取记录。

eg16-1.php 代码如下。

```
<?php
  echo "<pre>";
  $mysqli = new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr = "select * from tb_tbl";
  $mysqli_result = $mysqli->query($sqlstr);
  //执行 query()方法的 select 语句并返回 mysqli_result 类的对象
  $row1 = $mysqli_result->fetch_array();          //读取第一条记录
  //与$row1 = $mysqli_result->fetch_array(MYSQL_BOTH)等价
  var_dump($row1);                                //数组的 key 既包含数字又包含字段名
```

```
  echo "<br />";
  $row2 = $mysqli_result->fetch_array(MYSQL_ASSOC);   //读取第二条记录
  var_dump($row2);
  echo "<br />";                                      //数组的 key 只包含字段名
  $row3 = $mysqli_result->fetch_array(MYSQL_NUM);     //读取第三条记录
  var_dump($row3);                                    //数组的 key 只包含数字
  $mysqli_result->close();
  $mysqli->close();
?>
```

eg16-2.php 代码如下。

```
<?php
  $mysqli = new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr = "select * from tb_tbl";
  $mysqli_result = $mysqli->query($sqlstr);
  while ( $row = $mysqli_result->fetch_array(MYSQL_NUM)){  //循环读取全部记录
    //保存到二维数组$data 中，便于后面使用
    $data[] = $row;
  }
  /* 在现实程序设计中，还可以把$data 数组赋值给 session，这样在其他页面中
    也可以使用这些查询数据，避免频繁从远程服务器中读取数据 */
  for($i=0;$i<count($data);$i++){
    for($j=0;$j<count($data[0]);$j++)
      echo $data[$i][$j]."  ";
    echo "<br />";
  }
  $mysqli_result->close();
  $mysqli->close();
?>
```

4．fetch_object()方法

fetch_object()方法是以对象的形式返回结果集，而不是数组，因此返回的结果要以对象的方式进行访问。

【示例 17】 从数据库 db_db1 的表 tb_tbl 中读取记录。

eg17.php 代码如下。

```
<?php
  $mysqli = new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr = "select * from tb_tbl";
  $mysqli_result = $mysqli->query($sqlstr);
  while ( $row = $mysqli_result->fetch_object()){     //循环读取全部记录
    echo $row->id . " " . $row->user . " "; //字段前不要加$符号
    echo $row->password . " " . $row->yuwen . " ";
    echo $row->sex . " " . $row->addr . "<br />" ;
  }
  $mysqli_result->close();
  $mysqli->close();
```

```
?>
```

12.5.4 从结果集中获取数据列信息

数据集中通常包含多个数据列,mysqli_result 类提供了多个属性和方法用于获取数据集的列信息。

fetch_field()方法:获取结果集中当前列信息;

fetch_fields()方法:获取结果集中全部列信息;

field_count 属性:获取结果集中列数量;

current_field 属性:获取当前列位置;

field_seek()方法:改变当前列位置偏移量。

1. fetch_field()方法

fetch_field()方法用于从结果集中读取当前字段信息。返回当前字段信息,是对象类型。并将指针指向下一个字段。若不能读取当前字段信息,则返回 false。

【示例 18】 从数据库 db_db1 的表 tb_tbl 中读取一个字段信息。

eg18. php 代码如下。

```
<?php
  echo "<pre>";
  $mysqli = new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr = "select * from tb_tbl";
  $mysqli_result = $mysqli->query($sqlstr);
  $fieldinfo = $mysqli_result->fetch_field();
  var_dump($fieldinfo);
  $mysqli_result->close();
  $mysqli->close();
?>
```

程序运行结果为:

```
object(stdClass)#3 (13) {
  ["name"]=>
  string(2) "id"
  ["orgname"]=>
  string(2) "id"
  ["table"]=>
  string(6) "tb_tbl"
  ["orgtable"]=>
  string(6) "tb_tbl"
  ["def"]=>
  string(0) ""
  ["db"]=>
  string(6) "db_db1"
  ["catalog"]=>
  string(3) "def"
  ["max_length"]=>
  int(1)
```

```
  ["length"]=>
  int(4)
  ["charsetnr"]=>
  int(63)
  ["flags"]=>
  int(49667)
  ["type"]=>
  int(3)
  ["decimals"]=>
  int(0)
}
```

【示例 19】 从数据库 db_db1 的表 tb_tb1 中获得全部字段名。在用户不知道一个数据表的字段名时或字段是否变化时非常有用。

eg19. php 代码如下。

```
<?php
  echo "<pre>";
  $mysqli = new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr = "select * from tb_tb1";
  $mysqli_result = $mysqli->query($sqlstr);
  while ($fieldinfo = $mysqli_result->fetch_field()){
    $field_array[] = $fieldinfo->name;
  }
  var_dump($field_array);
?>
```

程序运行结果为：

```
array(6) {
  [0]=>
  string(2) "id"
  [1]=>
  string(4) "user"
  [2]=>
  string(8) "password"
  [3]=>
  string(5) "yuwen"
  [4]=>
  string(3) "sex"
  [5]=>
  string(4) "addr"
}
```

【示例 20】 获取数据库 db_db1 的 tb_tb1 表中全部数据信息并显示在表格中。在用户不知道一个数据表的字段名或字段数时使用 fetch_field()方法非常有用。

eg20. php 代码如下。

```
<?php
  $mysqli = new MySQLi('127.0.0.1','root','root','db_db1','3306');
```

```
    $mysqli->set_charset("utf8");
    $sqlstr = "select * from tb_tbl";
    $mysqli_result = $mysqli->query($sqlstr);
    echo "<table border='1' width='400'>";
    echo "<tr>";
    while ($fieldinfo = $mysqli_result->fetch_field()){
      echo "<td>".$fieldinfo->name."</td>";          //获得并输出结果集全部字段名
    }
    echo "</tr>";
    while($row = $mysqli_result->fetch_row()){
      echo "<tr>";
      for ( $i=0 ; $i<$mysqli_result->field_count ; $i++)   //field_count 字段数
        echo "<td>".$row[$i]."</td>";
      echo "</tr>";
    }
    echo "</table>";
    $mysqli_result->close();
    $mysqli->close();
?>
```

程序运行结果如图 12-32 所示。



id	user	password	yuwen	sex	addr
1	liubei	123456	78	male	wuhan
2	guanyu	abcdef	78	male	changsha
4	zhaoyun	qwertyui	98	female	tianjin
5	machao	23456789	80	male	nanjing
6	huangzhong	zxcvbnma	89	female	guangzhou
7	liubei	asdfghjk	97	male	chengdu

图 12-32　使用 fetch_field()方法获得全部字段信息

2. fetch_fields()方法和 field_count 属性

fetch_fields()方法用于从结果集中读取全部字段信息，返回结果为索引数组，每个数组元素是一个表示字段信息的对象。

field_count 属性表示结果集包含的字段的个数。

【示例 21】 从数据库 db_db1 的表 tb_tbl 中获得全部字段信息。

eg21. php 代码如下。

```
<?php
    echo "<pre>";
    $mysqli = new MySQLi('127.0.0.1','root','root','db_db1','3306');
    $mysqli->set_charset("utf8");
    $sqlstr = "select * from tb_tbl";
    $mysqli_result = $mysqli->query($sqlstr);
    $fieldinfo = $mysqli_result->fetch_fields();
    var_dump($fieldinfo);     //一个数组,每个元素是一个表示字段信息的对象
    $mysqli_result->close();
```

```
  $mysqli->close();
?>
```

【示例 22】 获取数据库 db_db1 的 tb_tb1 表中全部数据信息并显示在表格中。在用户不知道一个数据表的字段名或字段数时还可以使用 fetch_fields()方法以及 field_count 属性。

eg22.php 代码如下。

```
<?php
  echo "<pre>";
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr ="select * from tb_tb1";
  $mysqli_result =$mysqli->query($sqlstr);
  $fieldinfo =$mysqli_result->fetch_fields();
  //获得全部字段信息,是一个索引数组
  echo "<table border='1' width='400'>";
  echo "<tr>";
  for($i=0;$i<count($fieldinfo);$i++)  //count($fieldinfo)表示数组元素的个数
    echo "<td>".$fieldinfo[$i]->name."</td>";
    //对象数组$fieldinfo 的 name 属性表示字段名
  echo "</tr>";
  while($row =$mysqli_result->fetch_array()){
    echo "<tr>";
    for ($i=0;$i<$mysqli_result->field_count;$i++)  //field_count 结果集字段数
      echo "<td>".$row[$i]."</td>";
      //可以写成“echo "<td>".$row[$fieldinfo[$i]->name]."</td>";”
      //$fieldinfo[$i]->name 是字段名
    echo "</tr>";
  }
  echo "</table>";
  $mysqli_result->close();
  $mysqli->close();
?>
```

程序运行结果和示例 20 运行结果完全相同。

3. field_seek()方法和 current_field 属性

field_seek()方法用于将结果集中的指针设定为指定的字段偏移量。current_field 属性用于获得当前的字段指针偏移量。field_seek()方法语法格式为:

```
field_seek($fieldnr)
```

field_seek()方法用于将结果集中的指针设定为指定的字段偏移量。

说明:$fieldnr 参数就是要设置的字段指针偏移量,为正整数,从 0 开始,0 表示第 1 个字段,……

返回值:设置偏移量成功则返回 true,否则返回 false。

【示例 23】 field_seek()方法和 current_field 属性举例。用于获得结果集字段名。

eg23.php 代码如下。

```
<?php
  echo "<pre>";
  $mysqli = new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr = "select * from tb_tbl";
  $mysqli_result = $mysqli->query($sqlstr);
  $offset0 = $mysqli_result->current_field;  //当前偏移量为 0
  $field1 = $mysqli_result->fetch_field();   //第一个字段,即'id'
  $offset1 = $mysqli_result->current_field;
  //当前偏移量为 1,因为执行 fetch_field()方法后已经指向第 2 个字段了
  var_dump($offset0,$offset1,$field1);
  $bool = $mysqli_result->field_seek(3);     //3 表示要指向第四个字段,即'yuwen'
  var_dump( $bool );                         //为 TRUE,表示设置成功
  $fieldn = $mysqli_result->fetch_field();   //读出的是'yuwen'字段
  var_dump($fieldn);
  $mysqli_result->close();
  $mysqli->close();
?>
```

12.6 结构化查询语言

结构化查询语言(Structured Query Language)简称 SQL,是一种用于特殊目的的编程语言,也是一种数据库查询和程序设计语言,用于存取数据以及查询、更新和管理关系数据库系统,同时也是数据库脚本文件的扩展名。

结构化查询语言是高级的非过程化编程语言,允许用户在高层数据结构上工作。它不要求用户指定对数据的存放方法,也不需要用户了解具体的数据存放方式,所以具有完全不同于底层结构的不同数据库系统,可以使用相同的结构化查询语言作为数据输入与管理的接口。结构化查询语言语句可以嵌套,这使它具有极大的灵活性和强大的功能。

1986 年 10 月,美国国家标准协会对 SQL 进行规范后,以此作为关系式数据库管理系统的标准语言(ANSI X3. 135—1986),1987 年在国际标准组织的支持下成为国际标准。不过各种通行的数据库系统在其实践过程中都对 SQL 规范作了某些编改和扩充。所以,实际上不同数据库系统之间的 SQL 不能完全相互通用。

在 PHP + MySQL 编程中最常用的语句是 SELECT、INSERT INTO、DELETE、UPDATE、ALERT 和 CREATE 等语句。使用这些语句可以执行查询表记录、汇总与统计数据、插入记录、删除记录、修改记录、修改表结构和新建表等操作。

12.6.1 查询记录——SELECT 语句

SELECT 语句是功能非常强大的结构化查询语句,可以对记录进行筛选,对字段进行查询,还可以进行各种统计和汇总。

SELECT 语句的完整语法为:

```
SELECT[ALL|DISTINCT|DISTINCTROW|TOP]
```

```
{ * |talbe. * |[table.]field1[AS alias1][,[table.]field2[AS alias2][,...]]}
FROM tableexpression[,...][IN externaldatabase]
[WHERE...]
[GROUP BY...]
[HAVING...]
[ORDER BY...]
[WITH OWNERACCESS OPTION]
```

说明：用中括号([])括起来的部分表示是可选的，用大括号({})括起来的部分表示必须从中选择其中的一个。

- FROM 子句：FROM 子句指定了 SELECT 语句中字段的来源。FROM 子句后面是包含一个或多个的表达式(由逗号分开)，其中的表达式可为单一表名称、已保存的查询或由 INNER JOIN、LEFT JOIN 或 RIGHT JOIN 得到的复合结果。如果表或查询存储在外部数据库，在 IN 子句之后指明其完整路径。
- ALL、DISTINCT、DISTINCTROW、TOP 谓词：
  - ✍ ALL 返回满足 SQL 语句条件的所有记录。如果没有指明这个谓词，默认为 ALL。例如：

```
SELECT ALL FirstName,LastName FROM Employees
```

  - ✍ DISTINCT 表示如果有多条记录的选择字段的数据相同，只返回一个。
  - ✍ DISTINCTROW 表示如果有重复的记录，只返回一个。
  - ✍ TOP 显示查询头尾若干记录。也可返回记录的百分比，这是要用 TOP *N* PERCENT 子句(其中 *N* 表示百分比)。
- 用 AS 子句为字段取别名：如果想为返回的列取一个新的标题，或者经过对字段的计算或总结之后产生了一个新的值，希望把它放到一个新的列里显示，则用 AS 子句。
- WHERE 子句指定查询条件。
  - ✍ 比较运算符及其含义如下。
    - ✍ =：等于。
    - ✍ >：大于。
    - ✍ <：小于。
    - ✍ >=：大于等于。
    - ✍ <=：小于等于。
    - ✍ <>：不等于。
    - ✍ !>：不大于。
    - ✍ !<：不小于。
  - ✍ 范围(BETWEEN 和 NOT BETWEEN)，BETWEEN...AND...运算符指定了要搜索的一个闭区间。
  - ✍ 列表(IN，NOT IN)。IN 运算符用来匹配列表中的任何一个值。IN 子句可以代替用 OR 子句连接的一连串的条件。
  - ✍ 模式匹配(LIKE)。LIKE 运算符检验一个包含字符串数据的字段值是否匹配一

指定模式。LIKE 运算符里使用的通配符及其含义如下。

✍? 任何一个单一的字符。

✍* 任意长度的字符。

✍# 0～9 的单一数字。

✍[字符列表] 在字符列表里的任一值。

✍[! 字符列表] 不在字符列表里的任一值。

✍- 指定字符范围，两边的值分别为其上下限。

• 用 ORDER BY 子句排序结果。ORDER 子句按一个或多个（最多 16 个）字段排序查询结果，可以是升序（ASC）也可以是降序（DESC），默认是升序。ORDER 子句通常放在 SQL 语句的最后。ORDER 子句中定义了多个字段时，则按照字段的先后顺序排序。

• 分组和总结查询结果。在 SQL 的语法里，GROUP BY 和 HAVING 子句用来对数据进行汇总。GROUP BY 子句指明了按照哪几个字段来分组，而将记录分组后，用 HAVING 子句过滤这些记录。ROUP BY 子句的语法：

```
SELECT fidldlist FROM table WHERE criteria [ GROUP BY groupfieldlist [ HAVING
groupcriteria]]
```

• 聚集函数的意义：

✍SUM() 求和。

✍AVG() 平均值。

✍COUNT() 表达式中记录的数目。

✍COUNT(*) 计算记录的数目。

✍MAX() 最大值。

✍MIN() 最小值。

✍VAR() 方差。

✍STDEV() 标准误差。

✍FIRST() 第一个值。

✍LAST() 最后一个值。

为了讲述下面的示例，现有数据库 db_db1，数据库中有表 tb_tb1 和 tb_tb1，如图 12-33 所示。

id	user	password	yuwen	sex	addr
1	liubei	123456	90	male	wuhan
2	guanyu	abcdef	88	male	changsha
3	zhangfei	654321	56	male	Beijing
4	zhaoyun	qwertyui	78	female	tianjin
5	machao	23456789	76	male	nanjing
6	huangzhong	zxcvbnma	56	female	guangzhou
7	jiangwei	asdfghjk	67	male	chengdu
8	zhangbao	zxcvbnmq	91	female	chongqing

id	user	phoneno
1	liubei	13100000000
2	guanyu	13200000000
3	zhangfei	13300000000
4	zhaoyun	13400000000
5	machao	13500000000
6	huangzhong	13600000000
7	jiangwei	13700000000
8	zhangbao	13800000000

图 12-33 数据库 db_db1 中的表 tb_tb1 和 tb_tb2

【示例 24】 查询表 tb_tb1 中 yuwen（语文）最高分、最低分、平均分、不及格人数。

eg24. php 代码如下。

```
<?php
  echo "<pre>";
  $mysqli = new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr1 = "SELECT MAX(`yuwen`) AS MAX,MIN(`yuwen`)
  AS MIN,AVG(`yuwen`) AS AVG FROM tb_tbl";
  $mysqli_result1 = $mysqli->query($sqlstr1);
  $row1 = $mysqli_result1->fetch_array();
  echo '最高分: ' . $row1['max'].',最低分:' . $row1['min'] . ',平均分: ' . round($row1['avg
  '],2);
  $sqlstr2 = "SELECT COUNT(*) AS bjg FROM tb_tbl WHERE `yuwen`< 60" ;
  $mysqli_result2 = $mysqli->query($sqlstr2);
  $row2 = $mysqli_result2->fetch_array();
  echo ',不及格人数:'.$row2['bjg'];
  $mysqli_result1->close();
  $mysqli_result2->close();
  $mysqli->close();
?>
```

程序运行结果如下。

```
最高分: 91,最低分: 56,平均分: 75.25,不及格人数:2
```

【示例 25】 查询表 tb_tbl 中 sex 为`male`的记录的`yuwen`的平均分，查询 user 中姓`zhang`的记录的信息。

eg25. php 代码如下。

```
<?php
  echo "<pre>";
  $mysqli = new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr1 = "SELECT `sex` , AVG(`yuwen`) AS AVG FROM tb_tbl
  GROUP BY `sex` HAVING `sex`= 'male'";
  $mysqli_result1 = $mysqli->query($sqlstr1);
  $row1 = $mysqli_result1->fetch_array();
  echo $row1['sex'] . '--' . $row1['avg'] . "\n";
  $sqlstr2 = "SELECT * FROM tb_tbl WHERE `user` LIKE 'zhang%'";
  $mysqli_result2 = $mysqli->query($sqlstr2);
  while ( $row2 = $mysqli_result2->fetch_array() ){
    for($i = 0;$i < $mysqli_result2->field_count ; $i++)
      echo $row2[$i] . '   ';
    echo "\n";
  }
  $mysqli_result1->close();
  $mysqli_result2->close();
  $mysqli->close();
?>
```

程序运行结果如下。

```
male--75.4000
3   zhangfei   654321   56   male   Beijing
8   zhangbao   zxcvbnmq   91   female   chongqing
```

【示例 26】 查询表 tb_tbl1 中`yuwen`最高分同学的`phoneno`,其中`phoneno`在表 tb_tb2 中。

eg26. php 代码如下。

```
<?php
  echo "<pre>";
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr ="SELECT `user`,`phoneno` FROM tb_tb2 WHERE `user` =
  (SELECT `user` FROM tb_tbl1 WHERE `yuwen` = (SELECT MAX (`yuwen`) FROM tb_tbl1))";
  $mysqli_result =$mysqli->query($sqlstr);
  $row =$mysqli_result->fetch_array();
  echo $row['user'].'---'.$row['phoneno'];
  $mysqli_result->close();
  $mysqli->close();
?>
```

程序运行结果如下。

```
zhangbao---13800000000
```

13.6.2 插入记录——INSERT INTO 语句

INSERT INTO 语句用于向表格中插入新的行。语法格式为:

```
INSERT INTO 表名称 VALUES (值 1, 值 2,...)
```

或

```
INSERT INTO table_name (列 1, 列 2,...) VALUES (值 1, 值 2,...)
```

若不指定列名,则默认为全部列名,即全部字段。若指定列名,则值列表和列的列表要一一对应,且保证值的数据类型与列的数据类型相匹配。INSERT INTO 还可以一次性插入多条记录。插入成功返回 TRUE,插入失败返回 false。可以通过 mysqli 类的 affected_rows 属性来获得加插入成功的记录数。

【示例 27】 向表 tb_tbl1 中插入三条记录。

eg27. php 代码如下。

```
<?php
  echo "<pre>";
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr1 ="INSERT INTO  `tb_tbl`
  (`id` , `user` ,`password` ,`yuwen` ,`sex` ,`addr` )
  VALUES (NULL , 'guanxing', 'abcdef', '88', 'male', 'changsha');";
  if ($mysqli->query($sqlstr1))
```

```
    echo "成功插入记录条数为:".$mysqli->affected_rows;
  //id 自动加 1。sex 为默认值
  $sqlstr2 ="INSERT INTO `tb_tbl`
  ( `user` ,`password` ,`yuwen` ,`addr` )
  VALUES('sunquan', 'abcdef', '88', 'changsha'),
  ('yuanshao', '654321', '89','Beijing');";
  if ($mysqli->query($sqlstr2))
    echo "\n 成功插入记录条数为:".$mysqli->affected_rows;
  $mysqli->close();
?>
```

程序运行结果如下。

```
成功插入记录条数为:1
成功插入记录条数为:2
```

【示例 28】 向表 tb_tbl 中插入记录,插入的记录来自另外一张表 tb_tbl1bak。

eg28.php 代码如下。

```
<?php
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr1 ="INSERT INTO tb_tbl SELECT * FROM tb_tblbak";
  //向表 tb_tbl 中插入记录,插入的记录来自表 tb_tblbak,应确保 id 唯一
  if ($mysqli->query($sqlstr1))
    echo "插入的记录数为:".$mysqli->affected_rows;
  $mysqli->close();
?>
```

12.6.3　修改记录——UPDATE 语句

UPDATE 语句用于更新(修改)数据库表记录。update 语句的语法格式为:

```
UPDATE 表名 SET 字段 1=值 1[ ,字段 2=值 2[ , …]][ WHERE 条件 ]
```

该语句用于更新(修改)满足 WHERE 条件的记录,只修改满足条件记录的有些字段,即被列举出来的字段。在修改的过程中需要保证数据类型的匹配以及满足主键唯一等要求。

【示例 29】 使用 UPDATE 语句修改表 tb_tbl。

eg29.php 代码如下。

```
<?php
  echo "<pre>";
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr1 ="UPDATE tb_tbl SET `yuwen`=`yuwen`+5";  //修改全部记录的 yuwen
  if ($mysqli->query($sqlstr1))
    echo "修改的记录数为:".$mysqli->affected_rows;
  $sqlstr2 ="UPDATE tb_tbl SET `yuwen` = `yuwen` +5,`password`='12345678'
  WHERE `user` like 'zhang% '";
  //只修改 user 中姓 zhang 的记录的 yuwen 和 password
```

```
  if ($mysqli->query($sqlstr2))
    echo "\n修改的记录数为:".$mysqli->affected_rows;
  $mysqli->close();
?>
```

12.6.4 删除记录——DELETE 语句

DELETE 语句用于删除表记录，语法格式为：

```
DELETE FROM 表名称 [ WHERE <条件>]
```

DELETE 语句用于删除表中全部满足 WHERE 条件的记录。如果没有 WHERE 条件，则表示删除全部记录。

【示例 30】 使用 DELETE 语句删除 tb_tb1 表中若干记录。

eg30.php 代码如下。

```
<?php
  echo "<pre>";
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  //删除姓 zhang 的记录
  $sqlstr1 ="DELETE FROM tb_tb1 WHERE `user` LIKE 'zhang% '";
  if ($mysqli->query($sqlstr1))
    echo "删除的记录数为:".$mysqli->affected_rows;
  //删除 yuwen 为 60~70 分的记录
  $sqlstr2 ="DELETE FROM tb_tb1 WHERE `yuwen` BETWEEN 60 AND 70";
  if ($mysqli->query($sqlstr2))
    echo "删除的记录数为:".$mysqli->affected_rows;
  //删除 yuwen 分为 80、85、90 的记录
  $sqlstr3 ="DELETE FROM tb_tb1 WHERE `yuwen` IN (80,85,90)";
  if ($mysqli->query($sqlstr3))
    echo "删除的记录数为!:".$mysqli->affected_rows;
  //删除全部记录
  $sqlstr4 ="delete from tb_tb1";
  if ($mysqli->query($sqlstr4))
    echo "删除的记录数为!:".$mysqli->affected_rows;
  $mysqli->close();
?>
```

12.6.5 新建表——CREATE 语句

CREATE 语句用于创建数据表。该命令语法在前面章节中已做介绍。

【示例 31】 使用 CREATE 语句创建数据库。

eg31-1.php 代码如下。

```
<?php
  echo "<pre>";
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1','3306');   //选择了数据库
  $mysqli->set_charset("utf8");
```

```
  $sqlstr = "
  CREATE TABLE  IF  NOT  EXISTS  `tb_tblbak` (
  //可以注明数据库,IF  NOT  EXISTS 表示不存在则创建,存在则不覆盖
  `id` INT( 4 ) NOT NULL AUTO_INCREMENT ,
  `user` VARCHAR( 20 ) NOT NULL ,
  `phoneno` VARCHAR( 11 ) NOT NULL ,
  PRIMARY KEY (`id` )
  ) ENGINE = MYISAM DEFAULT CHARSET = utf8; ";
  if ($mysqli->query($sqlstr))
     echo "新建表成功!";
  $mysqli->close();
?>
```

eg31-2. php 代码如下。

```
<?php
  echo "<pre>";
  $mysqli = new MySQLi('127.0.0.1','root','root');     //没有选择数据库
  $mysqli->set_charset("utf8");
  $sqlstr = "
  CREATE TABLE `db_db1`.`tb_tblbak1` (                 //选择了数据库
  `id` INT( 4 ) NOT NULL AUTO_INCREMENT ,
  `user` VARCHAR( 20 ) NOT NULL ,
  `phoneno` VARCHAR( 11 ) NOT NULL ,
  PRIMARY KEY (  `id` )
  ) ENGINE = MYISAM DEFAULT CHARSET = utf8; ";
  if ($mysqli->query($sqlstr))
     echo "新建表成功!";
  $mysqli->close();
?>
```

12.6.6　获得数据库的全部表——SHOW TABLES

SHOW TABLES 命令用于显示数据库中的全部表。通过 mysqli 类的对象执行 query()方法来返回 mysqli_result 类的对象,再通过 fetch_array()等方法获得一个或多个表。

【示例 32】　获得数据库 tb_tbl 中全部表,判断某数据表是否存在。

eg32. php 代码如下。

```
<?php
  echo "<pre>";
  $mysqli = new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr1 = "show tables";
  $mysqli_result = $mysqli->query($sqlstr1);
  while($row = $mysqli_result->fetch_array()){
    $table_array[] = $row[0];     //$row[0]为数据库 db_db1 中的表名
  }
  var_dump($table_array);         //$table_array 数组中存放着数据库 db_db1 中的全部表
  if ( in_array('tb_tbl',$table_array))
    echo "\n 数据表 tb_tbl 存在";
```

```
  else
    echo "\n 数据表 tb_tbl 不存在";
  $mysqli->close();
?>
```

12.6.7 修改表结构——ALTER TABLE

ALTER TABLE 语句用于修改表结构。该语句语法比较复杂,读者可以查找相关手册。但是仍然可以大致分为以下操作。

1. 删除列

```
ALTER TABLE table_name DROP col_name
```

2. 增加列

```
ALTER TABLE table_name ADD col_name INT NOT NULL  COMMENT '注释说明'
```

3. 修改列的类型信息

```
ALTER TABLE table_name CHANGE col_name new_col_name BIGINT NOT NULL  COMMENT '注释说明'
```

4. 重命名列

```
ALTER TABLE table_name CHANGE col_name new_col_name BIGINT NOT NULL  COMMENT '注释说明'
```

5. 重命名表

```
ALTER TABLE table_name RENAME new_col_name
```

6. 删除表中主键

```
Alter TABLE table_name drop primary key
```

7. 添加主键

```
ALTER TABLE sj_resource_charges ADD CONSTRAINT PK_SJ_RESOURCE_CHARGES PRIMARY KEY (resid,
resfromid)
```

8. 添加索引

```
ALTER TABLE sj_resource_charges add index INDEX_NAME (name);
```

9. 添加唯一限制条件索引

```
ALTER TABLE sj_resource_charges add unique emp_name2(cardnumber);
```

10. 删除索引

```
ALTER TABLE tablename drop index emp_name;
```

当然不仅仅限于上面 10 种操作,限于篇幅不做一一介绍。此外上述操作可以同时进行,也就是说可以在一个命令中同时完成多项操作。

【示例 33】 修改 tb_tbl 的表结构。

eg33.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli = new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr = "ALTER  TABLE `tb_tbl`
             ADD `phoneno` VARCHAR( 11 ) NOT NULL,
             DROP `yuwen`";
  //添加`phoneno`,删除`yuwen`,可以同时进行
  if ($mysqli->query($sqlstr))
     echo "修改表结构成功!";
  //下面是显示修改表结构后的表结构
  $sqlstr = "show columns from tb_tbl";   //显示表结构
  $result = $mysqli->query($sqlstr);
  while ($row = $result->fetch_array())
    var_dump($row);                       //每次循环都可以详细地显示一个字段的信息
  $mysqli->close();
?>
```

【示例 34】 修改 tb_tbl 的表结构。

eg34.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli = new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr = "ALTER  TABLE `tb_tbl`
           CHANGE `phoneno` `phonenew` VARCHAR(11) NOT NULL";
  if ($mysqli->query($sqlstr))
     echo "修改字段名成功!";
  $sqlstr = "ALTER  TABLE `tb_tbl`
           rename `tb_table1`";
  if ($mysqli->query($sqlstr))
     echo "修改表名成功!";
  $mysqli->close();
?>
```

篇幅所限,不再一一举例,用户在对表结构进行修改时,可以考虑把多个操作分为多次完成,以免互相影响。

12.6.8　删除表——DROP TABLE

删除表是指删除整个表结构和数据,不可恢复。使用 DROP TABLE 命令可以删除表。语法格式为:

```
DROP TABLE 数据表名
```

删除一个不存在的表将会产生错误,如果在删除语句中加入 IF EXISTS 关键字可以避免此错误的发生。格式如下。

```
DROP TABLE IF EXISTS 数据表名
```

例如,下面的代码可以删除数据库 db_db1 中数据表 tb_tbbak:

```
$mysqli =new MySQLi('127.0.0.1','root','root','db_db1','3306');
$mysqli->query("drop table if exists tb_tbbak");
```

12.7 数据查询

数据查询是数据库操作中最常见的操作形式,是数据库操作的重点,虽然在前面的章节中已经涉及数据的基本查询。本节将继续讲解数据的较为复杂的查询操作。

12.7.1 字段查询

查询所有字段使用"*"来代替,查询指定字段可以列出这些字段,还可以为指定字段赋予别名。

【示例 35】 字段查询举例。

eg35.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr ="select * from tb_tbl ";
  if ($mysqli->query($sqlstr))
    echo "查询全部字段执行成功!\n";
  //此处省略显示记录的代码,读者可以自己完成
  $sqlstr ="select `user` as `xingming`,`yuwen`,`sex` from tb_tbl ";  //仅查询 3 个字段
  $result =$mysqli->query($sqlstr);
  while($row =$result->fetch_assoc()){
    $data[] =$row;
  }
  for($i=0;$i<$result->num_rows;$i++){
    echo $data[$i]['xingming'].'--'.$data[$i]['yuwen'].'--'.$data[$i]['sex']."\n";
    // $data[$i]['xingming']不能写成$data[$i]['user']
  }
  $mysqli->close();
?>
```

12.7.2 带 IN 关键字的查询

使用带 IN 关键字的查询可以判断某个字段的值是否在某集合中。如果该字段的值在集合中则满足查询要求,该记录将会被查询出来,否则不被查询出来。

语法格式为:

```
SELECT * FROM 表名 WHERE 条件 [NOT] IN (元素 1,元素 2,…,元素 n);
```

【示例 36】 带 IN 的查询。

eg36.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli = new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr = "SELECT * FROM tb_tbl WHERE `user` IN ('liubei','guanyu','machao')";
  $result = $mysqli->query($sqlstr);
  while($row = $result->fetch_row()){
    for($i=0;$i<$result->field_count;$i++)
      echo $row[$i]."  ";
    echo "\n";
  }
  $mysqli->close();
?>
```

12.7.3　带 BETWEEN AND 的范围查询

BETWEEN AND 关键字可以判断某个字段值是否在指定范围内。如果该字段值在指定范围内则满足条件，该记录将被查询出来，否则该记录不被查询到结果集。

语法格式为：

```
SELECT * FROM 表名 WHERE 条件[NOT] BETWEEN 值 1 AND 值 2
```

【示例 37】 带 BETWEEN AND 关键字的查询。

eg37.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli = new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr = "SELECT * FROM tb_tbl WHERE `yuwen` BETWEEN 80 AND 100";
  // $sqlstr = "SELECT * FROM tb_tbl WHERE `yuwen` NOT BETWEEN 70 AND 90";
  $result = $mysqli->query($sqlstr);
  while($row = $result->fetch_row()){
    for($i=0;$i<$result->field_count;$i++)
      echo $row[$i]."  ";
    echo "\n";
  }
  $mysqli->close();
?>
```

12.7.4　带 LIKE 的字符匹配查询

LIKE 是比较常见的比较运算符，使用 LIKE 可以实现数据的模糊查询。它有两种通配符：“%”和“_”。“%”可以匹配多个字符，它代表任意长度的任意字符，长度大于或者等

于 0，例如，wu%han 可以匹配以 wu 开始且以 han 结束的任意字符串；而 wu_han 则只能匹配以 wu 开始并以 han 结束，且二者之间只有一个字符的字符串。

【示例 38】 带 LIKE 关键字的查询。

eg38. php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr ="select * from tb_tbl where `user` like 'liu% '";
  // $sqlstr ="select * from tb_tbl where `user` like 'liu% ei'";
  //% 匹配 n(n≥0)个任意的字符
  // $sqlstr ="select * from tb_tbl where `user` like 'liu_ei'";
  //_匹配 1个任意字符
  $result =$mysqli->query($sqlstr);
  while($row =$result->fetch_row()){
    for($i=0;$i<$result->field_count;$i++)
      echo $row[$i]."  ";
    echo "\n";
  }
  $mysqli->close();
?>
```

12.7.5 带 IS NULL 关键字查询空值

IS NULL 用于判断字段的值是否为空值(NULL)。如果字段的值为空值，则满足查询条件，该记录将被查询出来，即进入查询结果集；若字段的值不为空值，则不满足查询条件，该记录不被查询出来，不进入查询结果集。其语法格式如下。

```
IS [NOT] NULL
```

其中 NOT 是可选参数，表示不是空值。

【示例 39】 带 IS NULL 关键字的查询。

eg39. php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr ="SELECT * FROM tb_tbl WHERE `sex` IS NOT NULL";          //显示 sex 不为空值的记录
  // $sqlstr ="SELECT * FROM tb_tbl WHERE `sex` IS NULL";           //显示 sex 为空值的记录
  $result =$mysqli->query($sqlstr);
  while($row =$result->fetch_row()){
    for($i=0;$i<$result->field_count;$i++)
      echo $row[$i]."  ";
    echo "\n";
  }
```

```
  $mysqli->close();
?>
```

12.7.6　带 AND 或 OR 的多条件查询

AND 关键字可以用于联合多个查询条件，只有同时满足多个条件时才会被查询出来并进入查询结果集。只要有一个条件不满足，则记录不会被查询出来。语法格式为：

```
条件 1 AND 条件 2 [ AND 条件 3 [...]]
```

OR 关键字可以用于联合多个查询条件，多个条件中只需满足一个或一个以上的条件就会被查询出来并进入查询结果集。只有全部条件都不满足才不会被查询出来。语法格式为：

```
条件 1 OR 条件 2 [OR 条件 3 [...]]
```

AND 和 OR 查询可以结合在一起，此时 AND 优先级高于 OR，可以使用括号改变优先级。

【示例 40】 带 AND 或 OR 关键字的查询。

eg40.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli = new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  //$sqlstr = "SELECT * FROM tb_tbl WHERE `sex`='male' AND `yuwen`>80";
  //查询 sex='male'且 yuwen>80 的记录
  $sqlstr = "SELECT * FROM tb_tbl WHERE `id`>3 OR `sex`='male' AND `yuwen`>80";
  //查询 id>3 的记录,此外还查询 sex='male'且 yuwen>80 的记录,AND 的优先级高于 OR
  $result = $mysqli->query($sqlstr);
  while($row = $result->fetch_row()){
    for($i=0;$i<$result->field_count;$i++)
      echo $row[$i]."  ";
    echo "\n";
  }
  $mysqli->close();
?>
```

12.7.7　用 DISTINCT 关键字去掉结果中的重复记录

使用 DISTINCT 关键字可以去除查询结果集中重复的记录。语法格式为：

```
SELECT DISTINCT 字段 1 [,字段 2 [...]] ...
```

如果有些记录的列表中给出的字段的值都相同，则去掉相同的记录，相同的记录中只保留最前面的那一条记录。

【示例 41】 带 DISTINCT 关键字的查询。

eg41.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr ="SELECT DISTINCT `user`,`sex` FROM tb_tb1";
  //如果多条记录的 user 和 sex 同时相同,则只保留最前面的那一条记录
  $result =$mysqli->query($sqlstr);
  while($row =$result->fetch_row()){
    for($i=0;$i<$result->field_count;$i++)
      echo $row[$i]."  ";
    echo "\n";
  }
  $mysqli->close();
?>
```

12.7.8 用 ORDER BY 关键字对查询结果进行排序

使用 ORDER BY 关键字可以对查询结果进行升序(ASC)和降序(DESC)排列,默认情况下 ORDER BY 按照升序输出结果。如果要按照降序排列,可以使用 DESC 来实现。语法格式为:

```
ORDER BY 字段名 1[ ASC | DESC ][ ,字段名 2[ ASC | DESC ][...] ]
```

其中,ASC 表示升序,因为是默认的,故可以省略不写;DESC 表示降序排列。如果是多个字段,则表示以字段 1 作为排序依据,字段 1 相同的再以字段 2 作为排序依据,以此类推。

【示例 42】 带 ORDER BY 关键字的查询。

eg42.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr ="SELECT `user`,`sex` FROM tb_tb1 ORDER BY `user`,`sex` desc";
  //按照 user 升序排列,user 值相同的再按照 sex 降序排列
  $result =$mysqli->query($sqlstr);
  while($row =$result->fetch_row()){
    for($i=0;$i<$result->field_count;$i++)
      echo $row[$i]."  ";
    echo "\n";
  }
  $mysqli->close();
?>
```

12.7.9 用 GROUP BY 关键字和 HAVING 关键字进行分组查询

通过 GROUP BY 查询可以将结果划分到不同的数组中,实现对记录进行分组查询。在分组计算时使用非常多,比如按照性别统计平均分,按照宿舍统计最高分,等等。

【示例 43】　带 GROUP BY 关键字的查询。

eg43-1. php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr ="SELECT `sex`,COUNT(*) AS `renshu` FROM tb_tbl GROUP BY `sex`";
  //按照 sex 来分组,统计 male 和 female 的人数
  $result =$mysqli->query($sqlstr);
  while($row =$result->fetch_row()){
    for($i=0;$i<$result->field_count;$i++)
      echo $row[$i]."  ";
    echo "\n";
  }
  $mysqli->close();
?>
```

eg43-2. php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr ="SELECT `sex`,COUNT(*) AS `renshu` FROM tb_tbl
  GROUP BY `sex` HAVING `sex`='male'";
  //按照 sex 来分组,统计 male 的人数
  $result =$mysqli->query($sqlstr);
  while($row =$result->fetch_row()){
    for($i=0;$i<$result->field_count;$i++)
      echo $row[$i]."  ";
    echo "\n";
  }
  $mysqli->close();
?>
```

HAVING 用于对分组的结果再进行筛选。

12.7.10　用 LIMIT 关键字的记录数量限制查询

在查询数据时,可能会查询出很多记录。而用户需要的记录可能只是一小部分,这样就需要来限制查询的结果。LIMIT 是 MYSQL 中一个比较特殊的关键字,它用来控制输出的行数。LIMIT 有两种格式。

1. LIMIT *m*

表示取查询结果中的 m 条记录,如果不足 m 条记录,则取出全部记录。

2. LIMIT *m*,*n*

表示从序号为 m 的位置开始取记录,取 n 条记录,不足 n 条则取出全部记录。

【示例 44】 带 LIMIT 关键字的查询。

eg44. php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli = new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  //$sqlstr = "SELECT *  FROM tb_tbl LIMIT 3";
  //查询前 3 条记录
  $sqlstr = "SELECT *  FROM tb_tbl LIMIT 3,2";
  //查询从第 4 条记录开始的连续 2 条记录,序号从 0 开始
  $result = $mysqli->query($sqlstr);
  while($row = $result->fetch_row()){
    for($i=0;$i<$result->field_count;$i++)
      echo $row[$i]."  ";
    echo "\n";
  }
  $mysqli->close();
?>
```

12.7.11 聚合函数查询

聚合函数的最大特点是它们能根据一组数据求一个值。聚合函数的结果值只根据选定行中非 NULL 的值进行行计算,NULL 将被忽略。

1. COUNT()函数

COUNT()函数对于除"*"以外的任何参数,返回所选结果集中的非 NULL 值的行的数目。对于参数"*"则返回全部行的数目。

【示例 45】 使用聚合函数 COUNT()统计各种性别的人数。

eg45. php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli = new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  $sqlstr = "SELECT `sex`,COUNT(*) AS `renshu` FROM tb_tbl GROUP BY `sex`";
  //按照 sex 来分组统计人数
  $result = $mysqli->query($sqlstr);
  while($row = $result->fetch_row()){
    for($i=0;$i<$result->field_count;$i++)
      echo $row[$i]."  ";
    echo "\n";
  }
  $mysqli->close();
?>
```

2. SUM()函数

SUM()函数可以用于求出表中某个字段(或表达式)取值的总和。SUM()函数还可以

配合 GROUP 进行分组统计。

【示例 46】 使用聚合函数 SUM()求某个字段的和。

eg46.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  //$sqlstr ="SELECT sum(`yuwen`) AS `zongfen` FROM tb_tbl";
  //统计全班学生 yuwen(语文)总分
  $sqlstr ="SELECT `sex`,SUM(`yuwen`) AS `zongfen` FROM tb_tbl GROUP BY `sex`";
  //按照 sex 来分组,统计各种 sex 的 yuwen(语文)分数之和
  $result =$mysqli->query($sqlstr);
  while($row =$result->fetch_row()){
    for($i=0;$i<$result->field_count;$i++)
      echo $row[$i]."  ";
    echo "\n";
  }
  $mysqli->close();
?>
```

3. AVG()函数

AVG()函数用于求某个字段(或表达式)的平均值,还可以配合 GROUP 使用,对分组的行进行求平均值。

【示例 47】 使用聚合函数 AVG()求某个字段的和。

eg47.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  //$sqlstr ="SELECT AVG(`yuwen`) AS `pingjunfen` FROM tb_tbl";
  //统计全班学生的 yuwen(语文)平均分
  $sqlstr ="SELECT `sex`,ROUND(avg(`yuwen`),2) AS `pingjunfen` FROM tb_tbl GROUP BY `sex`";
  //按照 sex 来分组,统计各种 sex 的 yuwen(语文)平均分,round 用于保留 2 位小数
  $result =$mysqli->query($sqlstr);
  while($row =$result->fetch_row()){
    for($i=0;$i<$result->field_count;$i++)
      echo $row[$i]."  ";
    echo "\n";
  }
  $mysqli->close();
  echo "<script>window.location.href='wangzhi.php'</script>";
?>
```

4. MAX()函数和 MIN()函数

MAX()函数用于求某个字段(或表达式)的最大值,MIN()函数用于求某个字段(或表

达式)的最小值。这两个函数还可以配合 GROUP 进行分组操作。

【示例 48】 使用聚合函数 MAX()函数和 MIN()函数求最大值和最小值。

eg48.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli = new MySQLi('127.0.0.1','root','root','db_db1','3306');
  $mysqli->set_charset("utf8");
  //$sqlstr = "SELECT MIN(`yuwen`) AS `zuidifen` FROM tb_tbl";
  //统计全班学生 yuwen(语文)最低分
  $sqlstr = "SELECT `sex`,MAX(`yuwen`) AS `zuigaofen` FROM tb_tbl GROUP BY `sex`";
  //按照 sex 来分组,统计各种 sex 的 yuwen(语文)最高分
  $result = $mysqli->query($sqlstr);
  while($row = $result->fetch_row()){
    for($i=0;$i<$result->field_count;$i++)
      echo $row[$i]."  ";
    echo "\n";
  }
  $mysqli->close();
?>
```

12.7.12 连接查询

连接查询就是把不同表的记录连接到一起的最普遍的方法。

1. 内连接查询

内连接查询要求被连接的表都匹配,不匹配的行被排除。

【示例 49】 使用内连接查询,查询 tb_tb1 以及 tb_tb2 中 user、addr 和 phoneno 信息。已知 db_db1 数据库下的两张表 tb_tb1 和 tb_tb2 的记录,如图 12-34 所示。

id	user	password	yuwen	sex	addr
1	liubei	123456	78	male	wuhan
2	guanyu	abcdef	78	male	changsha
4	zhaoyun	qwertyui	98	female	tianjin
5	machao	23456789	78	male	nanjing
6	huangzhong	zxcvbnma	89	female	guangzhou
7	liubei	asdfghjk	97	male	chengdu

id	user	phoneno
1	liubei	13100000000
3	zhangfei	13300000000
4	zhaoyun	13400000000
5	machao	13500000000
6	huangzhong	13600000000
7	jiangwei	13700000000
8	zhangbao	13800000000

图 12-34 tb_tb1 表和 tb_tb2 表

eg49.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli = new MySQLi('127.0.0.1','root','root','db_db1');
  $mysqli->set_charset("utf8");
  $sqlstr = "SELECT tb_tb1.user,addr,phoneno FROM tb_tb1,tb_tb2
  WHERE tb_tb1.user = tb_tb2.user";
```

```
  //或$sqlstr ="SELECT tb_tbl.user,addr,phoneno FROM tb_tbl,tb_tb2
  //用“WHERE tb_tbl.user =tb_tb2.user";”效果一样
  $result =$mysqli->query($sqlstr);
  while($row =$result->fetch_row()){
    for($i=0;$i<$result->field_count;$i++)
      echo $row[$i]."  ";
    echo "\n";
  }
  $mysqli->close();
?>
```

程序运行结果如图 12-35 所示。

```
liubei  wuhan  13100000000
liubei  chengdu  13100000000
zhaoyun  tianjin  13400000000
machao  nanjing  13500000000
huangzhong  guangzhou  13600000000
```

图 12-35 内连接查询结果

从图 13-35 可以看出,查询结果是表 tb_tb1 以及表 tb_tb2 中匹配的记录,即公共的记录。

2. 外连接查询——左连接

左连接表示将左边表的全部记录和右边表进行连接组合,返回结果包含左边表的全部记录,左边表不匹配的部分,在右边表相应的列中添加 NULL 值。左连接使用 LEFT JOIN 关键字注明。左连接格式为:

```
SELECT 字段名列表 FROM 表 1 LEFT JOIN 表 2 ON 表 1.字段=表 2.字段
```

左连接筛选出表 1 中全部的记录。如果表 2 中找不到与之匹配的,则使用 NULL 来填充相应的字段。

【示例 50】 使用左连接显示 tb_tb1 中每个人的通信方式。

eg50. php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1');
  $mysqli->set_charset("utf8");
  $sqlstr ="SELECT tb_tbl.user,addr,phoneno FROM tb_tbl LEFT JOIN tb_tb2
  ON tb_tbl.user =tb_tb2.user";
  $result =$mysqli->query($sqlstr);
  while($row =$result->fetch_row()){
    for($i=0;$i<$result->field_count;$i++)
      echo $row[$i]."  ";
    echo "\n";
  }
  $mysqli->close();
?>
```

程序运行结果如图 12-36 所示。

从图 13-36 所示结果可以看出,运行结果包含了左表 tb_tb1 中全部的记录。即使右表

找不到与之匹配的记录,也会显示左表,相应的右表字段添加 NULL 值。

```
liubei  wuhan  13100000000
guanyu  changsha
zhaoyun  tianjin  13400000000
machao  nanjing  13500000000
huangzhong  guangzhou  13600000000
liubei  chengdu  13100000000
```

图 12-36　LEFT JOIN 连接效果

3. 外连接查询——右连接

右连接表示将右边表的全部记录和左边表进行连接组合,返回结果包含右边表的全部记录,右边表不匹配的部分,在左边表相应的列中添加 NULL 值。右连接使用 RIGHT JOIN 关键字注明。右连接格式为:

```
SELECT 字段名列表 FROM 表 1 RIGHT JOIN 表 2 ON 表 1.字段= 表 2.字段
```

右连接筛选出表 2 中全部的记录。如果表 1 中找不到与之匹配的,则使用 NULL 来填充相应的字段。

【示例 51】 使用右连接显示 tb_tb2 中每个人的通信方式。

eg51. php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1');
  $mysqli->set_charset("utf8");
  $sqlstr ="SELECT tb_tbl.user,addr,phoneno FROM tb_tbl RIGHT JOIN tb_tb2
  on tb_tbl.user =tb_tb2.user";
  $result =$mysqli->query($sqlstr);
  while($row =$result->fetch_row()){
    for($i=0;$i<$result->field_count;$i++)
      echo $row[$i]."  ";
    echo "\n";
  }
  $mysqli->close();
?>
```

```
liubei  wuhan  13100000000
liubei  chengdu  13100000000
    13300000000
zhaoyun  tianjin  13400000000
machao  nanjing  13500000000
huangzhong  guangzhou  13600000000
    13700000000
    13800000000
```

图 12-37　RIGHT JOIN 连接效果

程序运行结果如图 12-37 所示。

从图 12-37 中可以看出,运行结果集包含表 tb_tb2 中全部的记录。在表 1tb_tb1 中若找不到与之匹配的记录,则相应的字段添加以 NULL 值。

12.7.13　子查询

所谓子查询就是 SELECT 查询是另一个查询的附属。MySQL 还支持多级子查询,当遇到多层子查询时,遵循从最内层开始到最外层的原则。

1. 带 IN 关键字的子查询

当子查询的结果列只包含一个值时,比较运算符才适用。如果一个子查询的结果集是值的列表,这时比较运算符就必须使用 IN 来替换。IN 运算符可以检测结果集中是否存在特定的值,如果检测成功就执行外面的查询。

【示例 52】 查询 tb_tb1 中存在 phoneno 的全部记录。

eg52. php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1');
  $mysqli->set_charset("utf8");
  $sqlstr ="SELECT * FROM tb_tbl WHERE user IN (SELECT user FROM tb_tb2)";
  $result =$mysqli->query($sqlstr);
  while($row =$result->fetch_row()){
    for($i=0;$i<$result->field_count;$i++)
      echo $row[$i]."  ";
    echo "\n";
  }
  $mysqli->close();
?>
```

2. 带比较运算符的子查询

如果子查询的结果只是一个值,使用带比较运算符的子查询比较方便。常用的比较运算有大于(>)、大于等于(>=)、小于(<)、小于等于(<=)和不等于(! =)。

【示例 53】 查询 tb_tbl 中 yuwen(语文)在平均分及以上的记录。

分析:yuwen(语文)的平均分显然是一个值,这时适合使用比较运算符。eg53. php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1');
  $mysqli->set_charset("utf8");
  $sqlstr ="SELECT * FROM tb_tbl WHERE yuwen >= (SELECT AVG(yuwen) FROM tb_tbl)";
  $result =$mysqli->query($sqlstr);
  while($row =$result->fetch_row()){
    for($i=0;$i<$result->field_count;$i++)
      echo $row[$i]."  ";
    echo "\n";
  }
  $mysqli->close();
?>
```

3. 带 EXISTS 关键字的子查询

使用满足 EXISTS 关键字的子查询时,内层查询如果不为空,则执行外层查询,否则不执行外层查询。

【示例 54】 查询 tb_tbl 中 yuwen(语文)在平均分及以上的记录。

eg54. php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli =new MySQLi('127.0.0.1','root','root','db_db1');
```

```
    $mysqli->set_charset("utf8");
    // $sqlstr ="SELECT * FROM tb_tbl WHERE NOT EXISTS (SELECT * FROM tb_tb2 WHERE `id`=61)";
    // 如果 tb_tb2 中没有 id=61 的记录,则执行外层查询并显示 tb_tbl 中的全部记录
    $sqlstr ="SELECT * FROM tb_tbl WHERE EXISTS(SELECT * FROM tb_tb2 WHERE `id`=1)";
    //如果 tb_tb2 中有 id=1 的记录,则执行外层查询并显示 tb_tbl 中的全部记录
    $result =$mysqli->query($sqlstr);
    while($row =$result->fetch_row()){
      for($i=0;$i<$result->field_count;$i++)
        echo $row[$i]."  ";
      echo "\n";
    }
    $mysqli->close();
?>
```

本例中,只要 tb_tb2 中存在 id=1 的记录,内层查询有记录存在,则执行外层查询,否则不执行外层查询。

NOT EXISTS 的用法与 EXISTS 的用法刚好相反。

4. 带 ANY 关键字的子查询

ANY 关键字表示满足其中任意一个条件。使用 ANY 关键字时,只要满足内层查询结果中的任意一个,就要执行外层查询。

【示例 55】 查询 tb_tbl 中 yuwen 分在 male 和 female 平均分中的较小值以上的全部记录。

eg55. php 代码如下。

```
<?php
    header("content-type:text/html;charset=utf-8");
    echo "<pre>";
    $mysqli =new MySQLi('127.0.0.1','root','root','db_dbl');
    $mysqli->set_charset("utf8");
    $sqlstr ="SELECT * FROM tb_tbl WHERE `yuwen` >
    ANY (SELECT AVG(`yuwen`) FROM tb_tbl GROUP BY `sex`)";
    //显示 tb_tbl 中 yuwen 大于(男生和女生平均分中的较小值)的全部记录
    $result =$mysqli->query($sqlstr);
    while($row =$result->fetch_row()){
      for($i=0;$i<$result->field_count;$i++)
        echo $row[$i]."  ";
      echo "\n";
    }
    $mysqli->close();
?>
```

本例中,tb_tbl 中的任意记录只要它的 yuwen 大于 male 的平均分或者 female 的平均分(即两个平均分中的较小值),则需要查询出该记录。“> ANY”的意思是大于其中任意一个即查询出来。

5. 带 ALL 关键字的子查询

ALL 关键字表示满足所有条件。使用 ALL 关键字时,只有满足内层查询语句返回的所有结果才可以执行外部查询。

【示例 56】 查询 tb_tbl 中 yuwen 分在 male 和 female 平均分中的较大值以上的全部记录。

eg56.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli = new MySQLi('127.0.0.1','root','root','db_db1');
  $mysqli->set_charset("utf8");
  $sqlstr = "SELECT * FROM tb_tbl WHERE `yuwen` >
  ALL (SELECT AVG(`yuwen`) FROM tb_tbl GROUP BY `sex`)";
  //显示 tb_tbl 中 yuwen 大于(男生和女生平均分中的较大值)的全部记录
  $result = $mysqli->query($sqlstr);
  while($row = $result->fetch_row()){
    for($i=0;$i<$result->field_count;$i++)
      echo $row[$i]."  ";
    echo "\n";
  }
  $mysqli->close();
?>
```

本例中，tb_tbl 中的任意记录只要它的 yuwen 大于 male 的平均分和 female 的平均分(即两个平均分中的较大值)，则需要查询出该记录。“> ALL”的意思是大于其中全部结果时才查询出来。

12.7.14 表记录的分页查询

在 MySQL 中使用 LIMIT 可以进行分页查询。LIMIT 格式之一：

```
LIMIT m,n
```

则表示从序号为 *m* 开始取记录，取 *n* 个，不足 *n* 个记录的则取出全部记录。按照这样的思路，分页查询的代码如下。

```
SELECT <字段列表> FROM <表名> LIMIT (pageno-1) * pagesize , pagesize
```

假如将表中的记录分页显示，每页显示 pagesize 条记录，很显然上面的代码能够显示第 pageno 页的记录。也就是说 pageno 表示页号，pagesize 表示每页显示记录数。而总页数则可以通过计算得到，即 pagecount = ceil(recount/pagesize)，其中 reccount 表示记录总数。

【示例 57】 分页显示 tb_tbl 中的记录，要求每页显示 5 条记录。

分析：首先要得到 tb_tbl 中记录总数 reccount，再通过 reccount 和每页记录数 pagesize 来计算出页数 pagecount。

eg57.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  echo "<pre>";
  $mysqli = new MySQLi('127.0.0.1','root','root','db_db1');
  $mysqli->set_charset("utf8");
   $sqlstr = "SELECT * FROM tb_tbl";
  $mysqli->query($sqlstr);
  $reccount = $mysqli->affected_rows;                //总记录数
```

```
    $pagesize = 5;                                        //每页记录数
    $pagecount = ceil($reccount / $pagesize);            //页数,ceil()函数用于取大求整
    $pageno = @$_GET['pageno'] ? $_GET['pageno'] : 1;   //确定查询哪一页,默认为第 1 页
    $offset = ($pageno-1) * $pagesize;
     $sqlstr = "SELECT * FROM tb_tbl LIMIT $offset,$pagesize";   //分页查询代码
    $result = $mysqli->query($sqlstr);
    while($row = $result->fetch_row()){
      for($i=0;$i<$result->field_count;$i++)
        echo $row[$i]."  ";
      echo "\n";
    }
    $mysqli->close();
?>
```

在浏览器中输入网址 http://localhost/ch13/eg56.php 并运行页面,因为网址中不存在参数 pageno,则 $_GET['pageno']不存在,此时 $pageno 取值 1,显示第 1 页。

而在浏览器中输入网址 http://localhost/ch13/eg56.php?pageno=2 并运行页面,因为网址中存在 pageno,即 $_GET['pageno']=2,即 $pageno=2,此时显示第 2 页记录。

为了方便单击超链接时直接显示想要的页面,还可以在程序最后显示超链接,用于分页跳转。代码如下。

```
for($p=1;$p<=$pagecount;$p++)
  echo "<a href='eg57.php?pageno=$p'>$p</a>   ";
```

程序运行效果如图 12-38 所示。



图 12-38　分页查询效果

12.8　综 合 案 例

本案例讲述如何进行一键安装,然后再讲述在表格中进行增、删、改、查等操作。本案例将自动创建数据库 db_chengji,自动新建数据表 tb_cj,并自动添加 5 条记录。然后再通过界面进行增、删、改、查操作。

步骤 1　在文件夹 ch12 中新建文件夹 eg58,并创建 install.php 网页,实现管理系统的一键安装代码如下。

eg58/install.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  //下面是表单和安装按钮
  echo "<form name='form1' method='post' action=''>";
  echo "<input type='submit' name='button' value='install'>";
  echo "</form>";
  //if 用于判断是否单击了"安装"按钮
  if (isset($_POST['button'])){
    $db_name='db_chengji8';
    $mysqli=@new mysqli("127.0.0.1","root","root","$db_name");  //选择数据库
    //if 语句用于判断数据库 db_chengji 是否存在
    if (mysqli_connect_errno()) {                             //不是 0 则说明连接失败
        //下面的代码用于新建数据库 db_chengji
        $mysqli=new mysqli("127.0.0.1","root","root");           //不带数据库选项
        $sql1="CREATE DATABASE $db_name";
        if ($mysqli->query($sql1)){
          echo "<script>alert('数据库{$db_name}建立成功!');</script>";
        }
        //下面的代码用于新建数据表 tb_user
        $tb_name='tb_user';
        $sql2="CREATE TABLE `$db_name`.`$tb_name` (
           `id` bigint( 4 ) NOT NULL AUTO_INCREMENT ,
           `user` varchar( 11 ) CHARACTER SET utf8 NOT NULL ,
           `pwd` varchar( 11 ) CHARACTER SET utf8 NOT NULL ,
           PRIMARY KEY ( `id` )
           ) ENGINE=MYISAM DEFAULT CHARSET=utf8;";
        if ($mysqli->query($sql2)) {
          echo "<script>alert('数据表{$tb_name}新建成功!');</script>";
        }
        //下面的代码用于添加记录
        $sql3="INSERT INTO  `$db_name`.`$tb_name`
                 (`id`,`user`,`pwd`) VALUES
                 (NULL,  'liubei',  '123456'),
                 (NULL,  'guanyu',  'abcdef');";
        if ($mysqli->query($sql3)) {
          $count=$mysqli->affected_rows;
          echo "<script>alert('向表{$tb_name}添加了{$count}个记录!');</script>";
        }
        //下面的代码新建表 tb_cj
        $tb_name2='tb_cj';
        $sql4="CREATE TABLE  `$db_name`.`$tb_name2` (
                `id` INT( 2 ) NOT NULL AUTO_INCREMENT ,
                `xuehao` VARCHAR( 8 ) NOT NULL ,
                `xingming` VARCHAR( 8 ) NOT NULL ,
                `yuwen` INT( 3 ) NOT NULL ,
                PRIMARY KEY (  `id` )
                ) ENGINE=MYISAM DEFAULT CHARSET=utf8;";
        if ($mysqli->query($sql4)){
          echo "<script>alert('数据表{$tb_name2}新建成功!');</script>";
        }
        echo "<script>alert('成绩管理系统安装完毕!');</script>";
```

```
            //显示登录页面
            echo "<script>window.location.href='login.php';</script>";
        }
    }
?>
```

步骤 2 在文件夹 eg58 中创建 login. php 网页,实现管理系统的登录,代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  //下面是表单和登录界面
  echo "<form name='form1' method='post' action=''>";
  echo "用户名<input type='text' name='user'><br />";
  echo "密   码<input type='password' name='pwd'><br />";
  echo "<input type='submit' name='button' value='login'/>";
  echo "</form>";
  @session_start();
  //if 用于判断是否单击了"登录"按钮
  if (isset($_POST['button'])){
    $db_name = 'db_chengji8';
    $mysqli =@new mysqli("127.0.0.1","root","root","$db_name");  //选择数据库
    $user =$_POST['user'];
    $pwd =$_POST['pwd'];
    $result =$mysqli->query("select * from tb_user
      where `user`='".$user."' and `pwd`='".$pwd."'");
    if ( $result->num_rows !=1){
      echo "<script>alert('用户名或者密码错误!');</script>";
    }else{
      $_SESSION['user'] =$_POST['user'];
      $_SESSION['pwd'] =$_POST['pwd'];
      echo "<script>window.location.href='main.php'</script>";
    }
  }
?>
```

运行网页 login. php,显示效果如图 12-39 所示。



图 12-39 登录界面

步骤 3 登录成功会显示 main. php 页面,main. php 页面用于数据表的增、删、改、查,代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
```

```
@session_start();
if (!isset($_SESSION['user'],$_SESSION['pwd'])){
  //不允许直接运行 main.php 并强行去登录
  echo "<script>window.location.href='login.php'</script>";
}else{
  //再次验证用户名和密码是否正确,防止从别的页面跳转到 main.php
  $db_name='db_chengji8';
  $mysqli=@new mysqli("127.0.0.1","root","root","$db_name");
  $user=$_SESSION['user'];
  $pwd=$_SESSION['pwd'];
  $result=$mysqli->query("select * from tb_user
    where `user`='".$user."' and `pwd`='".$pwd."'");
  if ($result->num_rows==1){                    //验证为合法用户,下面显示表记录页面
    $result=$mysqli->query("select * from tb_cj");
    //获得字段数
    $fieldcount=$result->field_count;
    //下面的代码用于获取表 tb_cj 的全部字段名
    while ($f=$result->fetch_field()) $fieldnames[]=$f->name;
    echo "<form name='form1' method='post' action=''>";
    //下面的代码将表记录显示在表格中
    echo "<table align='center' width='400' border='1'>";
    //下面的代码显示标题行,即表 tb_cj 的字段名
    echo "<tr>";
    for($i=0;$i<$fieldcount;$i++)
      echo "<th>".$fieldnames[$i]."</th>";
    echo "<th>操作</th>";
    echo "</tr>";
    //下面的代码显示表记录
    while($row=$result->fetch_row()){
      echo "<tr>";
      for($i=0;$i<$fieldcount;$i++)
        echo "<td>".$row[$i]."</td>";
      $id=$row[0];
      echo "<td align='center'>";
      echo "<input type='submit' name='edit$id' value='edit' /> ";
      echo "<input type='submit' name='delete$id' value='delete' /></td>";
      echo "</tr>";
    }
    $cols=$fieldcount+1;
    echo "<tr><td colspan='$cols' align='center'>";
    echo "<input type='submit' value='add' name='add' />";
    echo "</td></tr>";
    echo "</table>";
    echo "</form>";
  }
  //其他代码……
}
?>
```

程序运行效果如图 12-40 所示。

步骤 4　在 main. php 中继续添加代码,用于判断是否单击了“添加”按钮 add,代码

```
@session_start();
if (!isset($_SESSION['user'],$_SESSION['pwd'])){
  //不允许直接运行 main.php 并强行去登录
  echo "<script>window.location.href='login.php'</script>";
}else{
  //再次验证用户名和密码是否正确,防止从别的页面跳转到 main.php
  $db_name='db_chengji8';
  $mysqli=@new mysqli("127.0.0.1","root","root","$db_name");
  $user=$_SESSION['user'];
  $pwd=$_SESSION['pwd'];
  $result=$mysqli->query("select * from tb_user
    where `user`='".$user."' and `pwd`='".$pwd."'");
  if ($result->num_rows==1){                    //验证为合法用户,下面显示表记录页面
    $result=$mysqli->query("select * from tb_cj");
    //获得字段数
    $fieldcount=$result->field_count;
    //下面的代码用于获取表 tb_cj 的全部字段名
    while ($f=$result->fetch_field()) $fieldnames[]=$f->name;
    echo "<form name='form1' method='post' action=''>";
    //下面的代码将表记录显示在表格中
    echo "<table align='center' width='400' border='1'>";
    //下面的代码显示标题行,即表 tb_cj 的字段名
    echo "<tr>";
    for($i=0;$i<$fieldcount;$i++)
      echo "<th>".$fieldnames[$i]."</th>";
    echo "<th>操作</th>";
    echo "</tr>";
    //下面的代码显示表记录
    while($row=$result->fetch_row()){
      echo "<tr>";
      for($i=0;$i<$fieldcount;$i++)
        echo "<td>".$row[$i]."</td>";
      $id=$row[0];
      echo "<td align='center'>";
      echo "<input type='submit' name='edit$id' value='edit' /> ";
      echo "<input type='submit' name='delete$id' value='delete' /></td>";
      echo "</tr>";
    }
    $cols=$fieldcount+1;
    echo "<tr><td colspan='$cols' align='center'>";
    echo "<input type='submit' value='add' name='add' />";
    echo "</td></tr>";
    echo "</table>";
    echo "</form>";
  }
  //其他代码……
}
?>
```

程序运行效果如图 12-40 所示。

步骤 4 在 main. php 中继续添加代码,用于判断是否单击了“添加”按钮 add,代码



图 12-40　显示表记录(暂未添加表记录)

如下。

```
<?php
  ...
  //下面的粗体字是添加的代码,功能是判断是否单击了 add 按钮
    if (isset($_POST['add'])){
      $_SESSION['add'] = '12345678';     //可以用于防止直接运行 add.php 页面
      echo "<script>window.location.href='add.php'</script>";
    }
  }
?>
```

新建页面 add.php,用于添加记录,add.php 的代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  @session_start();
  if (@$_SESSION['add'] != '12345678'){
    die('非法用户,禁止添加记录');
  }
  if (isset($_POST['save'])){
      $db_name = 'db_chengji8';
    $mysqli = @new mysqli("127.0.0.1","root","root","$db_name");
    $xuehao = $_POST['xuehao'];
    $xingming = $_POST['xingming'];
    $yuwen = $_POST['yuwen'];
    $tb_name = 'tb_cj';
    $sql = "INSERT INTO $tb_name(`xuehao`,`xingming`,`yuwen`)
           VALUES ('$xuehao','$xingming','$yuwen')";
    $result = $mysqli->query($sql);
    if ($result){
      echo "<script>alert('insert into succeed!');</script>";
      echo "<script>window.location.href='main.php';</script>";
    }
  }
?>
<form name="form1" method="post" action="">
  <table width="220" border="1" align="center">
    <tr>
      <td colspan="2" align="center">添加记录</td>
    </tr>
    <tr>
```

```
            <td>学号</td>
            <td><input type="text" name="xuehao" ></td>
        </tr>
        <tr>
            <td>姓名</td>
            <td><input type="text" name="xingming" ></td>
        </tr>
        <tr>
            <td>语文</td>
            <td><input type="text" name="yuwen" ></td>
        </tr>
        <tr>
            <td colspan="2" align="center">
             <input type="submit" name="save" value="save">
            </td>
        </tr>
    </table>
</form>
```

直接运行 add.php，会显示“非法用户，禁止添加记录”，且不会显示添加界面。若从 login.php 中输入正确的用户名和密码，登录到 main.php，再单击 add 按钮，显示效果如图 12-41 和图 12-42 所示。



图 12-41　单击 main.php 页面中的 add 按钮运行 add.php 页面



图 12-42　添加记录后返回到 main.php 页面的效果

步骤 5　在 main.php 中继续添加代码来删除记录。首先在 main.php 中添加代码，用于判断哪一个“删除”按钮被单击了，然后再删除该行记录。代码如下。

```
while($row=$result->fetch_row()){
    echo "<tr>";
    for($i=0;$i<$fieldcount;$i++)
        echo "<td>".$row[$i]."</td>";
    $id=$row[0];
    echo "<td align='center'>";
    echo "<input type='submit' name='edit$id' value='edit' /> ";
    echo "<input type='submit' name='delete$id' value='delete' /></td>";
    echo "</tr>";
    //下面的粗体字代码是添加的代码
    if (isset($_POST["delete$id"])){
```

```
        //说明$id记录所对应的记录行的“删除”按钮被单击,准备删除该行
        $result =$mysqli->query("delete from tb_cj where `id`='".$id."'");
        if ($result){
          echo "<script>alert('delete OK!');</script>";
          //页面刷新
          echo "<script>window.location.href='main.php';</script>";
        }
      }
    }
```

步骤 6 在 main.php 中添加代码实现修改记录功能,首先判断是否单击了某个 edit 按钮。若单击了某个 edit 按钮,则跳到 edit.php 页面来具体实现记录的修改。在 main.php 中添加的代码如下。

```
while($row =$result->fetch_row()){
  echo "<tr>";
  for($i=0;$i<$fieldcount;$i++)
    echo "<td>".$row[$i]."</td>";
  $id =$row[0];
  echo "<td align='center'>";
  echo "<input type='submit' name='edit$id' value='edit' /> ";
  echo "<input type='submit' name='delete$id' value='delete' /></td>";
  echo "</tr>";
  if (isset($_POST["delete$id"])){
    //说明$id记录所对应的记录行的 delete 按钮被单击,准备删除该行
    $result =$mysqli->query("delete from tb_cj where `id`='".$id."'");
    if ($result){
      echo "<script>alert('delete OK!');</script>";
      //页面刷新
      echo "<script>window.location.href='main.php';</script>";
    }
  }
  //下面的粗体字代码是新添加的,用于判断是否单击了某个 edit 按钮
  if (isset($_POST["edit$id"])){
    //说明$id记录所对应的记录行的 edit 按钮被单击,准备修改该行
    $_SESSION['edit'] ='abcdefg';
    $_SESSION['id'] =$id;
    echo "<script>window.location.href='edit.php';</script>";
  }
}
```

新建页面 edit.php,用于具体实现记录的修改。edit.php 代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  @session_start();
  if (@$_SESSION['edit'] !='abcdefg'){
    die('非法用户,禁止修改记录');
  }
  $db_name ='db_chengji8';
  $tb_name ='tb_cj';
```

```
    $id = $_SESSION['id'];
    $mysqli = @new mysqli("127.0.0.1","root","root","$db_name");
    $sql = "SELECT *  FROM $tb_name WHERE `id` = '$id'";
    $result = $mysqli->query($sql);
    $row = $result->fetch_row();
    if (isset($_POST['save'])){
      $xuehao = $_POST['xuehao'];
      $xingming = $_POST['xingming'];
      $yuwen = $_POST['yuwen'];
      $sql = "UPDATE tb_cj
             SET `xuehao`='$xuehao',`xingming`='$xingming',`yuwen`='$yuwen'
             WHERE `id` = '$id'";
      if ($mysqli->query($sql)){
        echo "<script>alert('update ok');</script>";
        echo "<script>window.location.href='main.php';</script>";
      }
    }
?>
<form name="form1" method="post" action="">
  <table width="220" border="1" align="center">
    <tr>
      <td colspan="2" align="center">修改记录</td>
    </tr>
    <tr>
      <td>学号</td>
      <td><input type="text" name="xuehao" value="<?php echo $row[1];?>" ></td>
    </tr>
    <tr>
      <td>姓名</td>
      <td><input type="text" name="xingming" value="<?php echo $row[2];?>"></td>
    </tr>
    <tr>
      <td>语文</td>
      <td><input type="text" name="yuwen" value="<?php echo $row[3];?>"></td>
    </tr>
    <tr>
      <td colspan="2" align="center">
       <input type="submit" name="save" value="save">
      </td>
    </tr>
  </table>
</form>
```

说明：设置表单元素 xuehao、xingming 和 yuwen 的 value 属性分别为 $row[1]、$row[2] 和 $row[3]的目的是显示原来记录的初始值。

经过整理后的 main.php 完整代码如下。

```
<?php
  header("content-type:text/html;charset=utf-8");
  @session_start();
  if (!isset($_SESSION['user'],$_SESSION['pwd'])){
```

```
    //不允许直接运行 main.php 强行去登录
    echo "<script>window.location.href='login.php'</script>";
}else{
    //再次验证用户名和密码是否正确，则防止从别的页面跳到 main.php
    $db_name='db_chengji8';
    $mysqli=@new mysqli("127.0.0.1","root","root","$db_name");
    $user=$_SESSION['user'];
    $pwd=$_SESSION['pwd'];
    $result=$mysqli->query("select * from tb_user
      where `user`='".$user."' and `pwd`='".$pwd."'");
    if ($result->num_rows==1){                    //验证为合法用户，则下面显示表记录页面
      $result=$mysqli->query("select * from tb_cj");
      //获得字段数
      $fieldcount=$result->field_count;
      //下面的代码用于获取表 tb_cj 的全部字段名
      while ($f=$result->fetch_field()) $fieldnames[]=$f->name;
      //使用表单，方便单击 add、delete 和 edit 按钮
      echo "<form name='form1' method='post' action=''>";
      //下面的代码将 tb_cj 表的记录显示在表格中
      echo "<table align='center' width='400' border='1'>";
      //下面的代码显示标题行，即 tb_cj 表的字段名
      echo "<tr>";
      for($i=0;$i<$fieldcount;$i++)
        echo "<th>".$fieldnames[$i]."</th>";
      echo "<th>操作</th>";
      echo "</tr>";
      //下面的代码显示表的记录
      while($row=$result->fetch_row()){
        echo "<tr>";
        for($i=0;$i<$fieldcount;$i++)
          echo "<td>".$row[$i]."</td>";
        $id=$row[0];
        //每一个记录行的最后一列显示 edit 和 delete 按钮
        echo "<td align='center'>";
        echo "<input type='submit' name='edit$id' value='edit' /> ";
        echo "<input type='submit' name='delete$id' value='delete' /></td>";
        echo "</tr>";
        //循环判断某个 delete 按钮是否被单击
        if (isset($_POST["delete$id"])){
          //说明$id 记录所对应的记录行的 delete 按钮被单击。下面的代码删除该行
          $result=$mysqli->query("delete from tb_cj where `id`='".$id."'");
          if ($result){                              //删除成功
            echo "<script>alert('delete OK!');</script>";
            //删除记录后页面需要刷新
            echo "<script>window.location.href='main.php';</script>";
          }
        }
        //循环判断某个 edit 按钮是否被单击
        if (isset($_POST["edit$id"])){
          //$_SESSION['edit']用于防止直接运行 edit.php 页面
          $_SESSION['edit']='abcdefg';
```

```
                //传递 id 到 edit.php 页面,以免修改到错误的记录行
                $_SESSION['id'] = $id;
                echo "<script>window.location.href='edit.php';</script>";
              }
            }
            $cols = $fieldcount+1;
            //下面的行用于显示 add 按钮
            echo "<tr><td colspan='$cols' align='center'>";
            echo "<input type='submit' value='add' name='add' />";
            echo "</td></tr>";
            echo "</table>";
            echo "</form>";
          }
          if (isset($_POST['add'])){
            //$_SESSION['add']用于防止直接运行 add.php 页面
            $_SESSION['add'] = '12345678';
            echo "<script>window.location.href='add.php'</script>";
          }
        }
      ?>
```

至此,成绩管理系统的一键安装、增加记录、修改记录、删除记录的功能已经基本实现,篇幅所限,查询功能读者可以自己完成。

12.9 习　题

一、填空题

1. 在 PHP 创建 mysqli 连接对象时使用________方法。
2. 获得连接错误的方法为________。
3. 关闭数据库连接使用________方法。
4. 设置数据库字符集用________命令。
5. 使用________方法来获得查询结果集 mysqli_result 的字段数。
6. 使用________方法来获得查询结果集 mysqli_result 的查询记录数。

二、选择题

1. 下列不属于 fetch_array()方法的参数是________。
   A. MYSQLI_ASSOC　　B. MYSQLI_NUM
   C. MYSQLI_BOTH　　D. MYSQLI_ROW
2. 关于 fetch_array()、fetch_row()和 fetch_assoc()的说法正确的是________。
   A. fetch_array()方法是 fetch_row()和 fetch_assoc()两个方法的结合
   B. fetch_assoc()方法是 fetch_row()和 fetch_array()两个方法的结合
   C. fetch_array()方法获得的是索引数组
   D. fetch_row()方法获得的数据的字段名表示键,字段内容表示值
3. 关于 mysqli 类的方法 query(),下列说法正确的是________。

A. 该方法只能返回 false 或 true

B. 该方法只能返回查询结果集，不会返回 false

C. 该方法只能实现查询，无法实现增加记录操作

D. 该方法可以实现增删改查等操作，既可以返回查询结果集，也能返回 true 或 false

三、程序设计题

若存在数据库 db_chengji，该数据库中存在表 tb_cj，表 tb_cj 的字段为 id、xuehao、xingming、yuwen、shuxue、yingyu、zongfen 和 pingjunfen。请完成下面的任务：

1. 编写代码，向表中添加若干记录。
2. 编写代码，查询 yuwen 分在 90 以上的记录。
3. 编写代码，查询姓“张”的同学的信息。
4. 编写代码求全部记录的总分，保存到字段 zongfen 中。
5. 计算全部记录的平均分，保存到字段 pingjunfen 中。
6. 编写程序求表 tb_cj 的字段数、记录数。
7. 编写程序求表 tb_cj 的全部字段名。
8. 编写程序删除 id 为 5 的记录。
9. 编写程序将 xingming 为 liubei 的记录修改为 xingming 为 liuxuande。
10. 查询 yingyu 分数在 60～90 的记录数。

部分习题参考答案

第 1 章

一、选择题

1. A　2. C　3. B

二、上机实习

2.

```
<?php
  echo "Hello,PHP!";
?>
```

第 2 章

一、填空题

1. “<? php” 和 “? >”
2. 4
3. #
4. echo()
5. /*

二、选择题

1. D　2. A　3. D　4. C　5. B
6. D

三、程序设计题

(略)

第 3 章

一、填空题

1. 布尔型、浮点型、整型、字符串型
2. global
3. (int)
4. +
5. string
6. &$b
7. wuhan、Changshab、$ab
8. unset()
9. const
10. true
11. 40
12. false
13. 5
14. 1 和 5

二、选择题

1. A　2. D　3. A　4. C　5. C
6. C　7. C

三、程序设计题

(略)

第 4 章

一、填空题

1. 跳转语句
2. switch...case 语句
3. continue
4. do...while
5. foreach
6. 就近原则

二、选择题

1. C　2. B　3. A　4. A　5. A
6. A　7. B　8. B

三、程序设计题

(略)

第 5 章

一、填空题

1. 用户自定义函数
2. round()
3. function
4. return
5. time()
6. date()
7. round()

8. max()

9. rand()

10. date()

二、选择题

1. A 2. D 3. B 4. A 5. A

6. B 7. D

三、程序设计题

（略）

第 6 章

一、填空题

1. 二维数组

2. range()

3. array_push()

4. array_unshift()

5. array_shift()

6. array_pop()

7. asort()或者 arsort()等

8. 索引数组

9. 关联数组

10. array_reverse()

二、选择题

1. A 2. D 3. B 4. B 5. C

6. A

三、程序设计题

（略）

第 7 章

一、填空题

1. .

2. <<<

3. substr()

4. rtrim()

5. rtrim()

6. preg_split()

7. /\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*/（备注：答案不唯一）

8. \+?(\d+)\(?\((\d+)\)?(\d+)\-?(\d+)（备注：答案不唯一）

9. ^http://[_a-zA-Z0-9-]+(.[_a-zA-Z0-9-]+)*$（备注：答案不唯一）

二、选择题

1. A 2. B 3. A 4. B 5. B

6. A 7. A

三、程序设计题

（略）

第 8 章

一、填空题

1. 继承

2. get_class

3. abstract

4. __destruct()

5. static

6. 抽象的

7. 这些接口中的全部抽象方法

8. 一，多

9. 抽象、抽象、抽象

10. 构造函数

二、选择题

1. C 2. D 3. C 4. A 5. C

6. B 7. C

三、程序设计题

（略）

第 9 章

一、填空题

1. POST

2. GET

3. name

4. name

5. $_POST['name']

6. password

二、选择题

1. A 2. C 3. A 4. B 5. C

三、程序设计题

（略）

第 10 章

一、填空题

1. setcookie('city','wuhan');
2. session.auto_start = 1
3. session_start()
4. setcookie()
5. session

二、选择题

1. A　2. D　3. D

三、程序设计题

(略)

第 11 章

一、填空题

1. filetype()
2. fileatime()
3. file_exists()
4. readdir()
5. fwrite()

二、选择题

1. B　2. A　3. A　4. C　5. D

三、程序设计题

(略)

第 12 章

一、填空题

1. mysqli 的构造方法
2. mysqli_connect_error()
3. close()
4. set_charset()
5. field_count
6. num_rows

二、选择题

1. D　2. A　3. D

三、程序设计题

(略)

参 考 文 献

[1] 王伟平,贺春雷. PHP+MySQL 网站开发入门与提高[M]. 北京:清华大学出版社,2014.

[2] 潘凯华. PHP 从入门到精通[M]. 北京:清华大学出版社,2014.

[3] 软件开发技术联盟. PHP+MySQL 开发实践 [M]. 北京:清华大学出版社,2013.

[4] 牟奇春,汪剑. PHP 动态网站开发项目教程[M]. 北京:人民邮电出版社,2017.

[5] 传智播客高教产品研发部. PHP+Ajax+jQuery 网站开发项目式教程[M]. 北京:人民邮电出版社,2016.

[6] 黑马程序员. PHP 基础案例教程[M]. 北京:人民邮电出版社,2017.

[7] 传智播客. PHP+MySQL 网站开发项目式教程[M]. 北京:人民邮电出版社,2016.

[8] 潘凯华,李慧,刘欣. PHP 经典编程 265 例[M]. 北京:清华大学出版社,2012.

[9] 潘凯华,等. PHP 典型模块精讲[M]. 北京:清华大学出版社,2012.